NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

DATA FUSION ALGORITHM FOR THE VESSEL TRAFFIC SERVICES SYSTEM: A FUZZY ASSOCIATIVE SYSTEM APPROACH

by

Thomas M. Ruthenberg

March, 1995

Thesis Advisor: Murali Tummala

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DATA FUSION ALGORITHM FOR THE VESSEL TRAFFIC SERVICES SYSTEM: A FUZZY ASSOCIATIVE SYSTEM APPROACH	5. FUNDING NUMBERS None
6. AUTHOR(S) Ruthenberg, Thomas M.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) USCG Electronics Engineering Center, Wildwood, New Jersey	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis develops an algorithm to fuse redundant radar observations caused by multiple radar coverage of a vessel. Additionally, the algorithm automates track history and improves position accuracy by incorporating differential Global Positioning System (GPS) reports. The algorithm uses fuzzy membership functions as a measure of correlation and a fuzzy associative system to determine which observations represent the same vessel. The use of the fuzzy associative system produces a computationally efficient algorithm. A track correlation technique is used to automate the track history thereby reducing operator workload. Results of tests based on computer simulation show that the fusion algorithm correctly correlates and fuses the radar observations.

14. SUBJECT TERMS Vessel Traffic Services System, Data Fusion, Fuzzy Logic, Membership Functions	15. NUMBER OF PAGES 110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

DATA FUSION ALGORITHM FOR THE VESSEL TRAFFIC SERVICES SYSTEM:A FUZZY ASSOCIATIVE SYSTEM APPROACH

Thomas M. Ruthenberg
Lieutenant, United States Navy
B.S.E.E., University of Wisconsin, 1987

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March, 1995

Author: Thomas M. Ruthenberg
Thomas M. Ruthenberg

Approved by: Murali Tummala
Murali Tummala, Advisor

Roberto Cristi
Roberto Cristi, Second Reader

Michael A. Morgan
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

This thesis develops an algorithm to fuse redundant radar observations caused by multiple radar coverage of a vessel. Additionally, the algorithm automates track history and improves position accuracy by incorporating differential Global Positioning System (GPS) reports. The algorithm uses fuzzy membership functions as a measure of correlation and a fuzzy associative system to determine which observations represent the same vessel. The use of the fuzzy associative system produces a computationally efficient algorithm. A track correlation technique is used to automate the track history thereby reducing operator workload. Results of tests based on computer simulation show that the fusion algorithm correctly correlates and fuses the radar observations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The Vessel Traffic Services (VTS) system is a surveillance and communications network designed to facilitate safe harbor navigation. Different versions of the system are used in many ports worldwide, to reduce operational and environmental risk to marine navigation. [Ref. 1] The system is an advisory service to inform pilots of hazardous conditions, such as navigation obstacles and other traffic. The system consists of multiple radars, video cameras and VHF radios for communication.

Currently, the U.S. Coast Guard operates eight VTS systems. The Coast Guard is proceeding on two fronts in the effort to improve the VTS systems: (1) development of the new VTS 2000 system and (2) upgrade of the existing VTS systems by improving software and equipment. Additionally, 15 ports are selected to be outfitted with a new VTS system and three ports with existing VTS systems will be retrofitted with the new system.

The VTS 2000 system is currently under development. The system will use commercially available components and employ an open architecture/open system design. [Ref. 2] The open architecture design will allow for upgrades as technology improves. This system also allows each port to modify the basic design as is necessary for that port's specific needs.

A. THESIS OBJECTIVE

The Coast Guard is currently upgrading and expanding the surveillance and communications systems of the VTS's at the New York, Puget Sound and San Francisco ports. [Ref. 3] This thesis proposes an algorithm to improve an existing VTS system, which is part of the Coast Guard's upgrade process. The algorithm uses fuzzy membership functions to correlate and fuse the redundant observations. The goals of this thesis are to remove redundant radar displays to reduce clutter, automate historical tracking information to reduce operator workload, incorporate Differential Global Positioning System (DGPS) inputs for improved position accuracy, and ensure that the algorithm is fast enough for real time processing.

B. THESIS OUTLINE

In the second chapter the hardware and software of a typical VTS system are described. This chapter describes the current operation of the system, demonstrating the need for the proposed algorithm. In Chapter III the concepts of data fusion are presented. Data fusion is used for tracking and removal of redundant radar track data in areas of overlapping radar coverage. Chapter IV presents the basics of fuzzy logic. To compare classical and fuzzy design techniques, a controller for the inverted pendulum problem is presented as an example. In Chapter V the proposed fusion algorithm is described and tested. A harbor traffic area, typical radar data and DGPS reports are simulated to evaluate the performance of the algorithm. Conclusions of the work presented and possible future direction are included in Chapter VI.

II. SYSTEM OVERVIEW

The VTS system is comprised of multiple Remote Site Subsystems (RSS) and a centralized Vessel Traffic Control Subsystem (VTCS). Each remote site provides vessel tracking, communication sensors and camera video for vessels operating within its area of responsibility (AOR). It is common for these AORs to overlap. The VTCS integrates, processes and displays the remote site data. The VTS system acquires data from up to 18 remote sites in the form of radar track data, digitized radar image data, camera video and audio. Additionally, the system acquires and processes DGPS data from reporting vessels. The data from the remote sites is transmitted via a microwave or a fiber optic communications link to the Vessel Traffic Center (VTC). The VTS system operates seven days a week, 24 hours a day, and is required to have a system availability of 99.9%. The algorithm in this thesis will reside in the sensor data processor at the VTC. An overview of the VTS system is shown in Figure 1.

A. REMOTE SITE SUBSYSTEM

Each VTS consists of a number of RSS which provide radar data, camera video and audio communicated to the VTC. The number of sites is determined by the geography of the port, capabilities of the radar being used and the size of the area selected by the Coast Guard for monitoring. A typical RSS configuration is composed of a radar unit, radar data processor, remote site processor, one to four video cameras and a VHF radio system. Components of the RSS are briefly described in the following.

1. Radar Transmitter/Receiver

The radar provides raw radar data for processing by the radar data processor. A typical radar is the Raytheon Pathfinder radar unit, in use at the VTS, New York. It is a surface search I-band radar with 25 kW power, pulse repetition rate of 3000/2000 PPS and pulse width 0.06 µsec (12 NM or less) / 0.5 µsec (24 NM or greater). The range

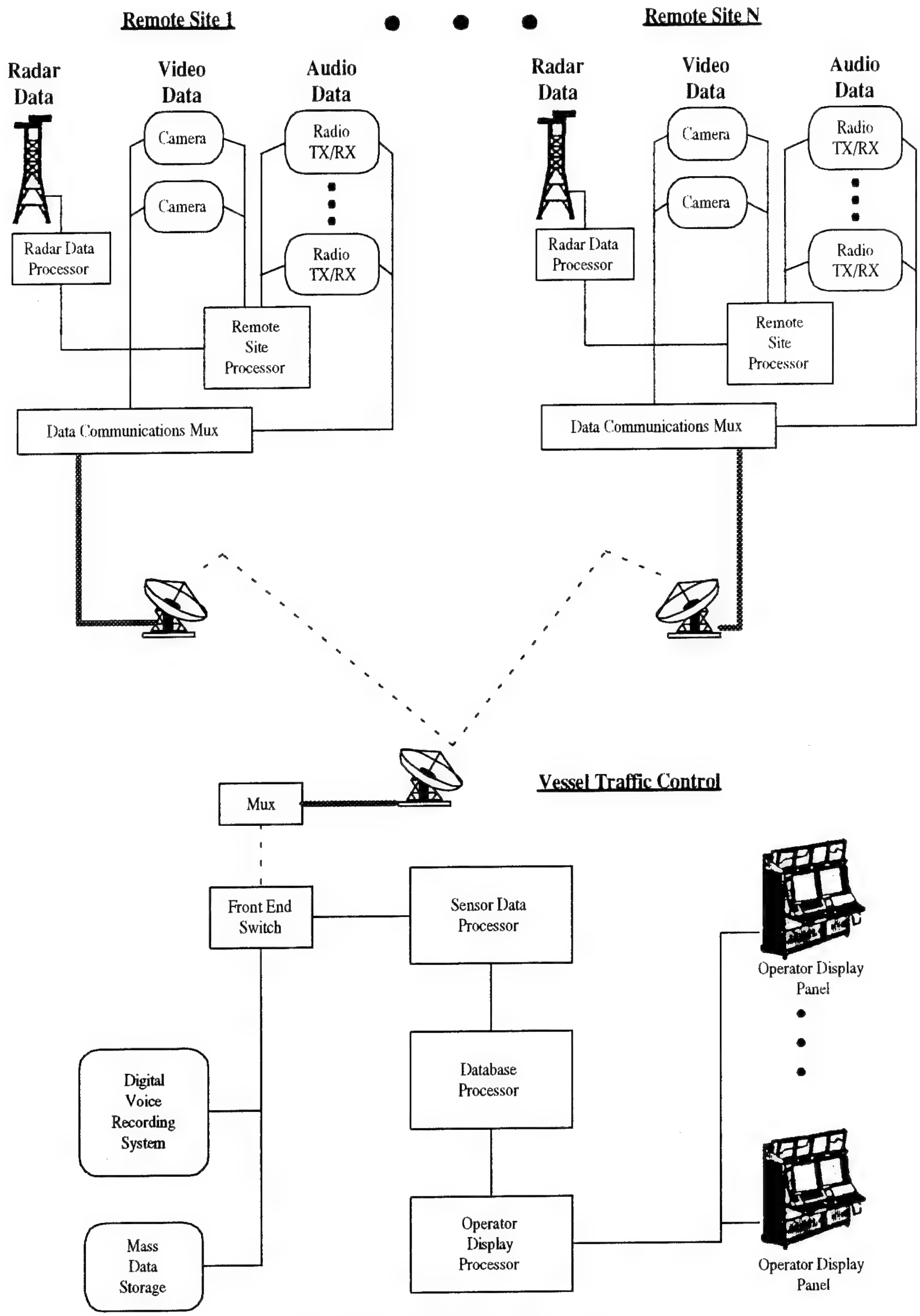


Figure 1 VTS System Overview

resolution of this radar is 9.84 yards at 3-12 NM, and the bearing resolution is specified as +/- 1° at 12 NM. Radar operating range is selectable and is typically set for a three to eight nautical mile maximum range.

2. Radar Data Processor

The radar data processor processes the raw radar data to generate radar tracks and run-length encoded image data. The track processing performed in this processor can also be called sensor level data fusion. The radar track updates from this processor are used by the data fusion algorithm proposed in this thesis. Proper functioning of the radar data processor is extremely important to the success of the data fusion algorithm.

The VTS system at New York currently uses a Folsum radar processor and will soon be upgrading to the Telephonics radar processor to obtain improved tracking capability. The Telephonics radar processor has been installed and is being evaluated at the VTS system at Puget Sound.

3. Remote Site Processor

The Remote Site Processor (RSP) provides an interface to control and monitor all remote site subsystems. RSS setup and control involves setting up and performing run-time control of system sensors including the radar processor, radar control unit, audio transceivers and video equipment consisting of cameras, pan/tilt units and video compression equipment where employed. [Ref. 4] Typically this processor routes operator control signals to position a camera or to change the power setting of a radar.

4. Video Cameras

A remote site may be configured with up to four cameras. Typically a remote site has a single black and white camera but may also have a color camera. The cameras are controlled by the operators at the VTC. Up to four cameras may be displayed at each operator display. The video display is very useful during daytime operation for monitoring

harbor traffic. During night operations, the video display is less effective but may be used to monitor vessel position lights.

5. VHF Radio System

Each remote site contains up to two VHF-FM transceivers and four guard transceivers. The frequencies and call signs for each VTS system are published in the Federal Radionavigation Plan (FRP). The radio system is used to broadcast harbor information continuously and to communicate as needed with vessel pilots.

B. VESSEL TRAFFIC CONTROL SUBSYSTEMS

The VTC processes, records and displays all incoming data from the remote sites. A VTC may consist of up to 16 display consoles. The major subsystems at the VTC are: Sensor Data Processor (SDP), Database Processor (DBP), visual image routing system, audio routing system and the Operator Display Processor (ODP). The algorithm proposed here will reside in the SDP. The various components of the VTC are briefly described below.

1. Sensor Data Processor

The SDP receives all incoming sensor data — radar image data, radar track data, video data, GPS data and radio communications. The purpose of the SDP is to perform track processing (fusion algorithm) and to distribute data to the DBP, ODP, mass data recorder and the digital voice recording system. The SDP also routes control signals to each remote site. These control signals are used to position cameras and set radar parameters.

2. Database Processor

The DBP currently maintains the historical record of each vessel transiting the harbor and performs alarm processing. The historical record includes: radar image data, radar track data, video camera data, alarm data and audio channel information. This algorithm will provide the identity information necessary for the historical radar track data. The DBP also performs alarm processing which is an important safety feature of the VTS system. The alarm

processing includes alarms for: vessel proximity to defined geographical areas, close proximity alarm to another vessel, swing circles for anchored vessels and moving safety zones.

3. Operator Display Processor

The ODP processor allows the operator to setup and control remote sites and to select data for display. Up to four video cameras may be chosen for display. Using the features of the ODP processor, the operator can display any combination of chart displays, concurrently. As a result the operator is able to view the entire VTS surveillance area as well as zoom-in on a particular sector simultaneously.

C. OPERATION OF A TYPICAL VTS

Currently, to generate a hardcopy of a vessel's path through a VTS equipped harbor requires operator inputs as the vessel transits in order to maintain path history. Additionally, redundant observations are not removed by the display algorithm, instead all radar targets are displayed from all reporting remote sites.

Operators select colors for vessel icons covered by each of the radars. All vessel icons being displayed from a given remote site are displayed with the same color. As a vessel transits from one radar coverage area to another, the vessel icon changes color. In the areas where the radar coverage overlaps, multiple icons of a vessel are displayed. Depending on the resolution of the display, the target may appear as one vessel or as multiple vessels close to one another. When the vessel changes color the operator must select the new radar, so the path history of that vessel can be continued. If a remote site is not selected, the display is not affected but the historical record of the vessel will not be continued beyond the previous radar.

Figure 2 shows a simulation of the VTS in New York harbor used for this thesis with a single vessel transiting northeast from the Bank Street remote site to the Governors Island remote site. The simulated radar coverage area for each radar is shown. Assume that the Bank Street remote site tracks are displayed using red and Governors Island tracks are

displayed in blue. As the vessel transits, a single red icon is displayed when the vessel is reported by the Bank Street remote site only. In the overlap region, two distinct vessels can sometimes be seen; this is caused by the radar measurement errors and true north misalignment of the individual radars. In the overlap region the operator either sees one vessel displayed in the color of the latter reporting remote site or two distinct red and blue icons in close proximity. In this region the operator must select the Governors Island site to provide historical track information. If the operator fails to select the Governors Island site to provide track information for the historical record, the track history of this vessel will end after the vessel departs the Bank Street radar coverage area. As the vessel continues, it departs the Bank Street coverage area and appears as a single blue vessel in the Governors Island coverage area.

The proposed fusion algorithm is designed to automate the tracking of vessels, freeing the operator from continually selecting remote sites to provide track data. The algorithm recognizes and fuses multiple observations of the same vessel and uses the fused data for display and historical recording. The algorithm also proposes a method of correcting the estimates provided by each radar based on differential GPS reports, utilizing the accuracy of differential GPS to improve radar position reports. Finally, computational efficiency has been considered at all levels of processing to reduce computational expense.

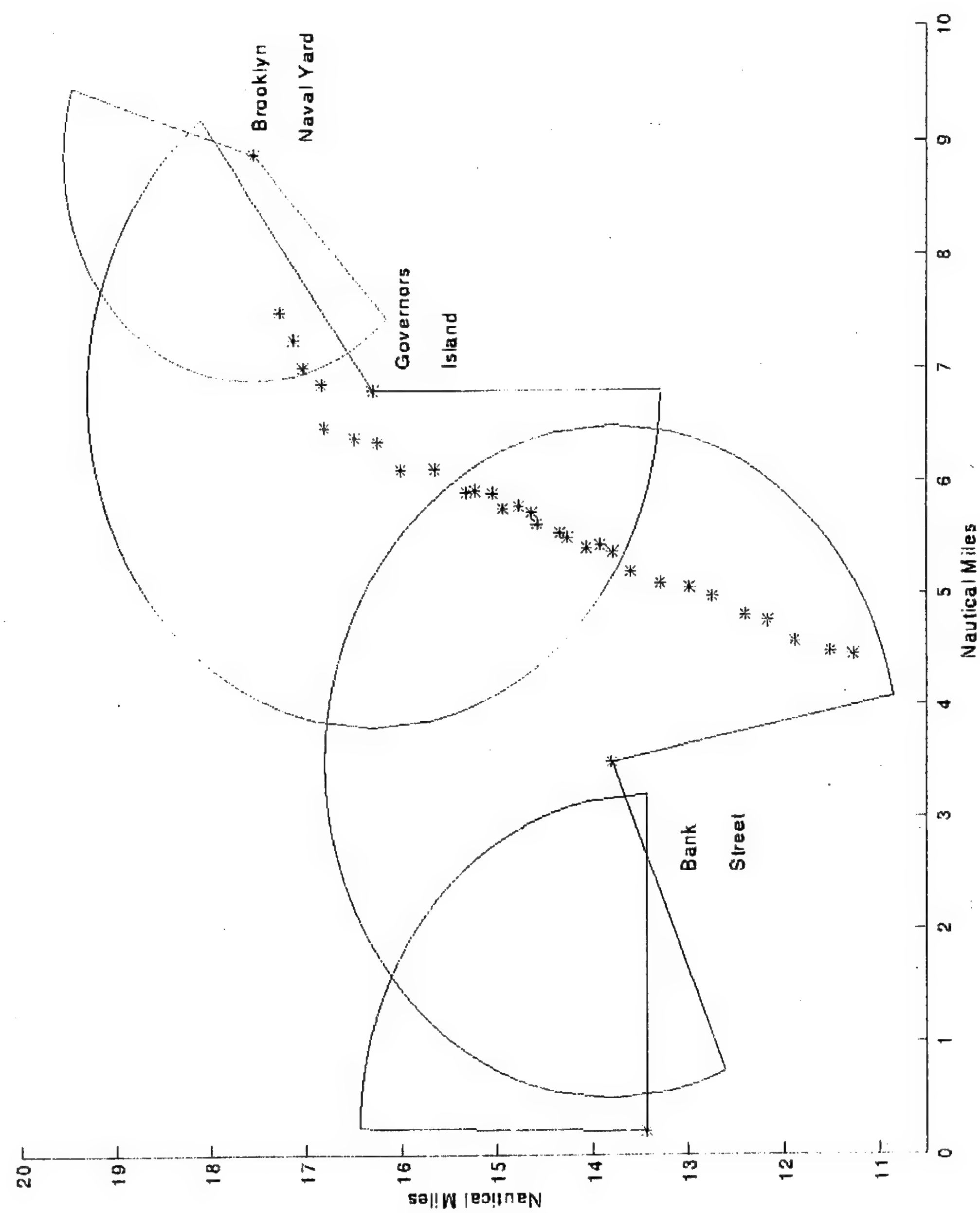


Figure 2 Typical VTS Operator Display With Redundant Observations

III. DATA FUSION

Data fusion is a process dealing with the association, correlation and combination of data from multiple sources to achieve a refined position and identity estimation. [Ref. 5] The goal of data fusion is to derive more information by combining data than is present in any single source. Humans perform data fusion by combining sight, sound, smell and touch to recognize objects in their environment. The enjoyment of a favorite food involves recognition of a distinctive texture, smell, appearance and flavor. Anti-submarine forces combine remotely sensed data such as sonar, voice reports and radar to detect submarines.

The fusion of multiple sensor data has many performance benefits for the problems of detection, tracking and identification. With multiple sensors a system is more robust; if a single sensor fails, the system can continue to operate. Spatial coverage is improved by fusing the data from sensors located at different sites. Detection is improved because where one sensor may not detect a target, a different type of sensor may detect the target. Improved confidence in the position and identity of a target is also a result of a multiple sensor environment.

Data fusion can be performed in three levels: positional fusion, identity fusion and threat assessment. [Ref. 6] Positional fusion seeks to determine an improved position estimate of a target by combining parametric data, such as azimuth, range, and range rate. Identity fusion uses known characteristics to determine target identification. [Ref. 5] Threat assessment is the highest level of data fusion and is used for military or intelligence fusion systems to determine the meaning of the fused data from an adversarial view. [Ref. 6] The algorithm this thesis proposes performs positional fusion; therefore, this type of fusion will be discussed in greater detail than identity or threat fusion.

A. POSITIONAL FUSION

Positional fusion is performed at two levels in the VTS system, sensor level and central level. Each remote site performs sensor level positional data fusion for tracking. The

remote site data fusion is performed in the radar data processor. The remote site tracking performed in the radar data processor is critical to the performance of the central level processing performed by the fusion algorithm because the algorithm has limited ability to check the data. The sensor level tracking updates vessel tracks, initiates new tracks and removes false tracks.

Central level fusion receives inputs from multiple sources. The incoming data is not necessarily of the same form. Central level data fusion may fuse data from radar, infrared detectors and electronic intelligence (ELINT) receivers. The proposed fusion algorithm performs central level fusion on radar tracks from multiple remote sites, GPS reports from reporting vessels and prospective list data entered by the Vessel Traffic Center operators. The central level processing is performed in the sensor data processor and will be discussed in detail in Chapter V.

Positional fusion seeks to determine a refined position estimate based on multiple observations of a target. A block diagram of positional fusion is provided in Figure 3. The terms in parenthesis in Figure 3 are specific to the central level processing performed by the fusion algorithm. The various operations of a typical positional fusion process are described in the following.

1. Sensors and Preprocessing

Sensors scan their respective surveillance volume and report all detected targets once every scan. At each remote site preprocessing is performed on the raw radar data. This preprocessing may include thresholding, filtering, clutter removal and analog to digital (A/D) or D/A conversion.

The sensor level data fusion process serves as preprocessing for central level fusion. It reduces the central level computation by performing the detection and estimation of tracks at the remote sites. Additionally, central level preprocessing is designed to filter the incoming data to control the rate of data into the fusion algorithm. Typically any fusion algorithm can be computationally overwhelmed by the data from multiple sources. Preliminary filtering may be designed to filter data according to reporting time, reporting

location, sensor type, and other characteristics, such as speed. Due to the update rate of the radar data processors and the relatively slow speeds of the vessels being tracked, more data is available than is necessary for display; therefore, in the proposed fusion algorithm filtering is performed by discarding many of the radar observations.

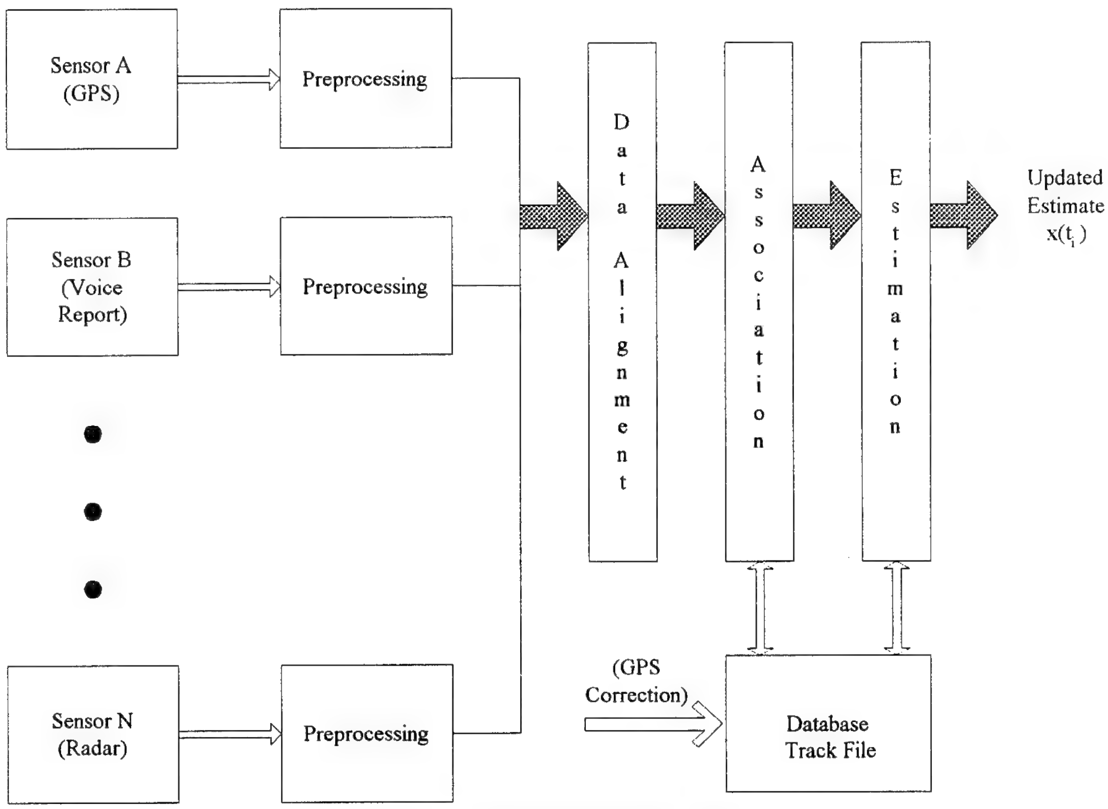


Figure 3 Positional Fusion [Ref. 6]

2. Alignment

Data alignment operation functions to orient the data from multiple sensors to common spatial and temporal references. Raw radar data, for example, must be converted from azimuth and range referenced to the reporting radar to a spatial reference common to all reporting radars such as longitude and latitude, or a local Cartesian coordinate system.

To associate data, the data must also be aligned to a common time. Temporal alignment is required when observations to be processed are not matched in time. This is a

common situation because sensors are not typically controlled to acquire data synchronously. Temporal alignment is performed by a variety of techniques. Common temporal alignment techniques used are the alpha-beta tracker and the Kalman filter. The alpha-beta tracker is a fixed coefficient, deterministic filter, and it does not require any *a priori* statistical information about the target being tracked or the tracking device; however, the most common method used for position estimation is the Kalman filter. This filter requires statistical assumptions about the target and the measurement device (sensor).

Assuming the target dynamics and measurement noise are accurately modeled, the Kalman filter is more responsive to target maneuvering than the alpha-beta tracker, as can be seen in Figure 4. The improved accuracy of the Kalman filter over that of the alpha-beta filter is due to the dynamically calculated Kalman gains. The residual or innovations is the difference between the actual and expected measurement. In Kalman filtering the residual is checked for consistency. Large residual values indicate a target maneuver; thus implying the need for larger (more responsive) gains. [Ref. 7] The alpha-beta coefficients are fixed prior to implementation; the coefficients should be set based on the target being tracked. Decreasing the coefficient values leads to a less responsive filter which will perform better to noisy inputs while increasing the coefficients leads to better performance with dynamic inputs. These filters are used to determine an estimated future position based on the historical track. The estimate is the predicted position at the next $(k+1)^{th}$ scan based on the smoothed track of observations up to the k^{th} scan. This estimated position is then used by the association portion of the data fusion algorithm.

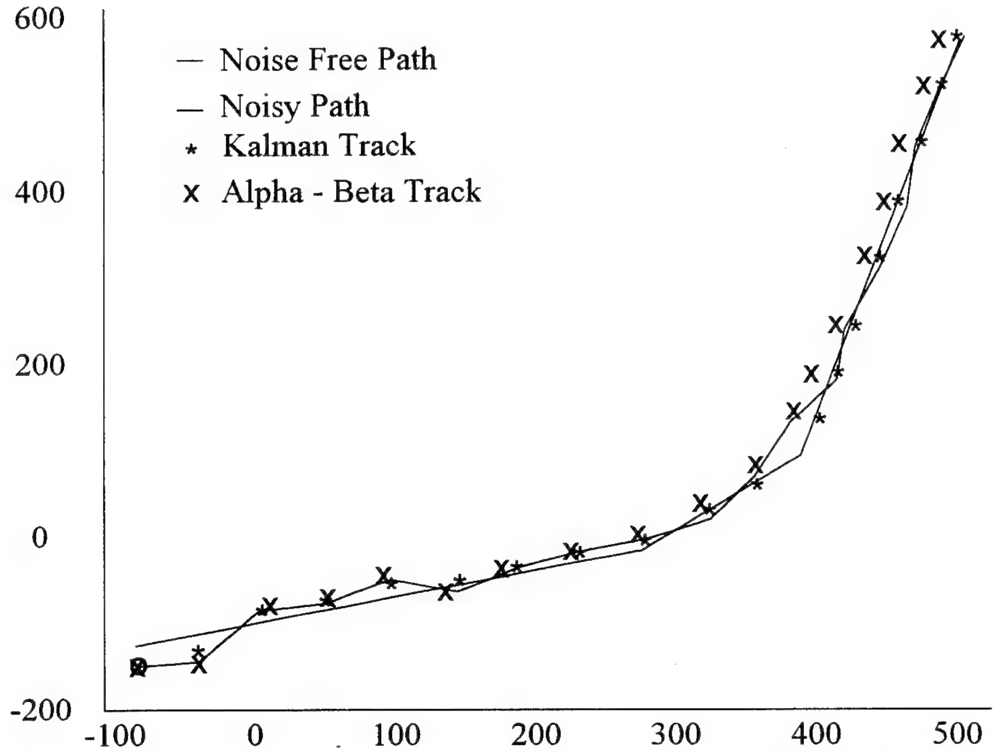


Figure 4 Comparison of Alpha-Beta Tracker and Kalman Filter

3. Association

Association seeks to group together observation pairs or observations to existing tracks. This is accomplished by evaluating the "closeness" of the match through some measure of correlation. Common measures of correlation are distance measures, fuzzy membership functions, correlation coefficients or some other figure of merit. These measures of correlation are also called association metrics. Gating can be used to reduce the number of observation pairs which must be analyzed. Gating techniques establish boundaries or limits to determine which pairs could physically be related, such as removing an observation from further consideration based on position or speed mismatch.

Associating observations to an existing track is performed at the completion of each scan in the remote site radar processor. The association process must determine whether each

observation is an update of an existing track, a new track, or a false track. An example of single sensor tracking is shown in Figure 5. The existing track is projected forward in time to the observation time (temporal alignment). Each observation is compared with the aligned estimates based on a measure of correlation, and the observation occurring within the allowable gate and association metric threshold is determined to be the update observation for that track. In this example observation one ($\triangle_1$) is the update for track one, observation two ($\triangle_2$) is the update for track two and observation three ($\triangle_3$) is a new or false track which will be determined on subsequent scans. Most tracking algorithms establish update criteria, such as 'three track updates in five scans' are required prior to establishing a new track, otherwise the observation is determined to be a false track. This example uses location information only to determine if an observation is an update to the track. To increase confidence that the correct observation is being chosen, additional parameters may be used to compute the association metric. An association metric based upon location, kinematics, measured target attributes or other parameters may be computed.[Ref. 4] The threshold used for the sensor level tracking association metric is dependent upon the update rate of the sensors, measurement error and state estimator performance. With a high update rate more stringent criteria may be established for association.

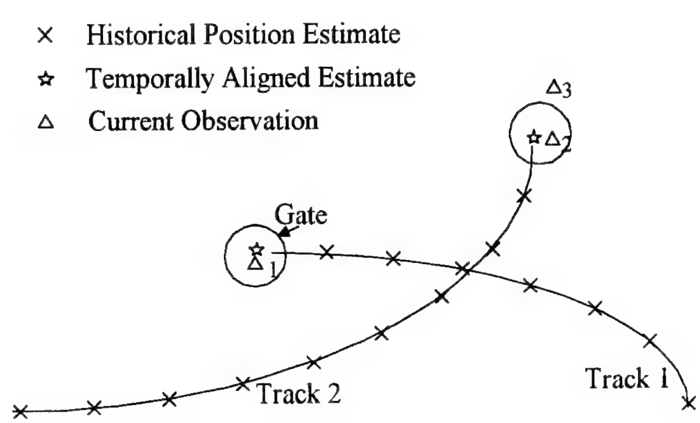


Figure 5 Sensor Level Data Fusion (Tracking)

The type of association metric to use is dependent on the available data. Distance measures are used to quantify the similarities between observations when only location information is available. Measures of correlation methods are used to determine similarities between vectors containing diverse information. The measure of correlation (ρ) is given as:

$$\rho = \frac{\sum_{i=1}^{N} \delta_i W_i \xi_i}{\sum_{i=1}^{N} \delta_i W_i} \tag{1}$$

where N is the number of parameters to be compared, W_i is a weight associated with each parameter, δ_i is a factor set to one if the parameter to be compared is present in both records and is set to zero otherwise, and ξ_i is the figure of merit, a functional measure of association that ranges from zero to one. [Ref. 6] The central level fusion proposed in this thesis uses fuzzy membership values as the figures of merit, and modifies the measure of correlation by using the minimum figure of merit instead of their weighted sum.

Gating techniques are used to reduce the computation time of the association process. An object's current position and *a priori* knowledge of its maximum speed or maximum turn rate may be used to eliminate unlikely observations from further consideration. The size of a gate is dependent on the maximum speed and maneuverability of the object being tracked. Observations within the gate will have all parameters evaluated for association; those outside the gate are eliminated from further association processing. In central level fusion, it may be that two sensor level tracks from different sensors have been associated repeatedly in the past; thus, if a track correlation history is maintained, track pairs that were previously established using more stringent association criteria can be re-established using less stringent requirements, such as satisfying a simple gating criterion. [Ref. 7]

4. Estimation

Estimation is the process of determining the state of the target based on the association of the observation pairs and the location estimate generated in the alignment process. As discussed in the alignment section, in sensor level fusion the output of the tracking filter is a smoothed track. The computation of the smoothed position by combining the estimated position at time k with the observation at time k is the estimation process. If the alpha-beta tracker is used, this filter assumes the smoothed position of a target is the weighted average of the predicted position and the new observation.[Ref. 5] The Kalman filter smoothed position estimate is a weighted (Kalman gain) sum of the estimate and the observation.

In central level fusion, *multiple* observations are selected by the association process as being the updates to the track. There are many techniques to determine the *best fit* of the observations to update the target state. The most familiar techniques are the least squares and weighted least squares methods. The least squares method minimizes the sum of squares of the error. In the weighted least squares method, error contributors may be selectively weighted. [Ref. 7] In systems which fuse data from different sensor types, the weight assigned is generally dependent on the accuracy of the sensor. In essence the observations with greater accuracy can be weighted to make a greater contribution to the solution.

Least squares methods cannot be used to solve under-determined systems. An under-determined system occurs when there are fewer observations than states. These systems may be solved by finding the centroid of each parameter. If equal weight is given to each parameter of the state vector, the centroid of a parameter is simply the average of that parameter from each observation. Centroidal fusion is used in the proposed fusion algorithm for state estimation of redundant observations.

B. IDENTITY FUSION

Identity fusion seeks to determine the identity of the target based on multiple observations of the target. Figure 6 shows the concept of single sensor identity declaration. Sensors used for identity declaration may provide imagery data or time series data.

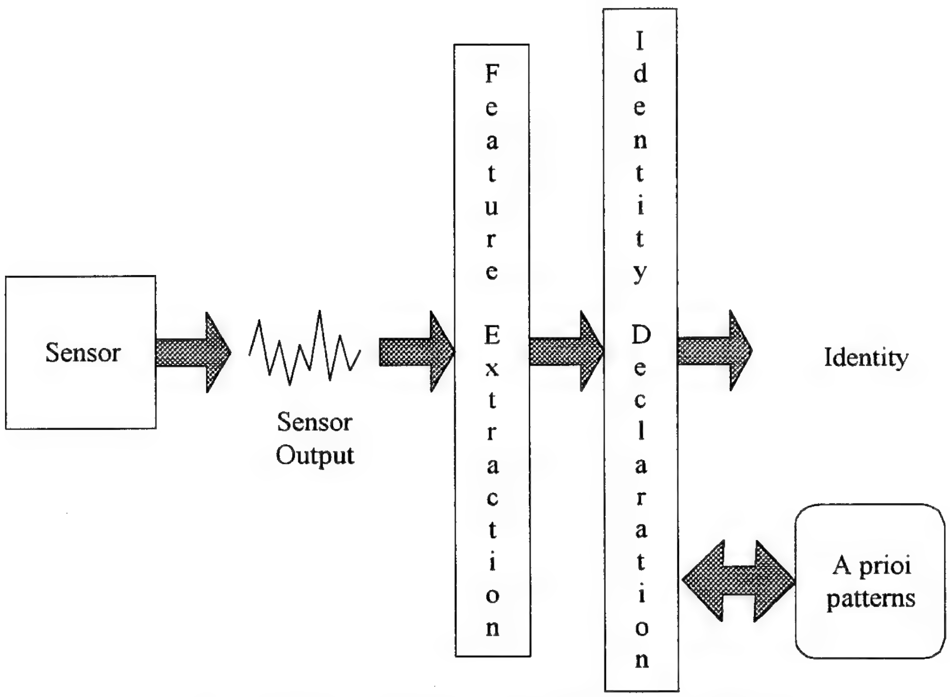


Figure 6 Concept of Identity Declaration [Ref. 6]

Feature extraction is performed to accurately represent the sensor output with a reduced number of data points. Feature extraction can be performed by numerous transformations to obtain a feature vector to represent the observed data. Common feature vector components for imagery data include discrete cosine transform coefficients, autoregressive (AR) model coefficients, aspect angle , shape, length and width. Features extracted from time series data include amplitude and location of spectral components, average signal amplitude, Fourier coefficients and AR model coefficients.

Identity declaration is analogous to the association process. The extracted feature vector is compared to the stored patterns to determine the identity. Techniques available for

identity declaration using the feature vector as an input include: adaptive neural networks, cluster algorithms and statistical pattern recognition. Each of these methods essentially requires the multi-dimensional input feature vector to be "close" to the stored pattern.

In the VTS system the radar image data could be used for identity fusion. These techniques are computationally expensive due to the size of the data being processed and are not considered viable for this application.

C. THREAT ASSESSMENT

The purpose of threat assessment processing is to determine the meaning of the fused data from an adversarial view. Threat assessment functions include determination of the lethality of friendly and enemy forces, assessment of force compositions, estimate of danger, targeting calculations and weapon assessment calculations. [Ref. 6]. These systems serve as an aid to military analysts and fuse information such as geographical data, enemy troop and equipment composition, prevailing level of hostilities and weather.

This chapter discussed the three levels of data fusion: positional, identity and threat assessment. Threat assessment is not applicable here and identity fusion is considered computationally too expensive for the proposed fusion algorithm. In the VTS system, the tracking performed by the remote site radar processors is considered sensor level positional fusion. The steps of positional fusion have been discussed using the sensor level tracking as an example. The development of the proposed algorithm will follow the steps (preprocessing, alignment, association and estimation) discussed here.

IV. FUZZY LOGIC

A. FUNDAMENTALS

In the following discussion classical set theory based logic is referred to as "crisp" logic and fuzzy set theory ideas as fuzzy logic. Fuzzy logic considers the partial relationship or membership of an object to a set. A membership function, $\mu(x)$, is used to grade the elements of a set to the interval [0,1]. The grade of membership can be considered a measure of the compatibility of an object to a defined set. The closer an object is graded to one, the higher the membership of the object in the set and the more compatible the object is with the set being considered.

The theory of fuzzy sets deals with a subset $\boldsymbol{A}$ in a universal set or universe of discourse $\boldsymbol{X}$. The universe of discourse, $\boldsymbol{X}$, is a set of objects whose elements are denoted by $\boldsymbol{x}$. A fuzzy subset has no well defined boundaries whereas the universe of discourse covers a definite range of objects. In classical set theory an element may either "belong to" the subset or "not belong to" the subset. A fuzzy subset allows for the possibility of partial membership. The fuzzy subsets are written as a set of ordered pairs

$$\boldsymbol{A} = \{x,\ \mu_A(x)\},\quad x \in \boldsymbol{X} \tag{2}$$

or

$$\boldsymbol{A} = \{\mu_A(x)/x\},\quad x \in \boldsymbol{X} \tag{3}$$

where $\mu_A(x)$ is the membership of x in $\boldsymbol{A}$. The membership value assigned is based on the membership functions mapping the universe of discourse. Every element in the universe of discourse has a membership associated with all the subsets described within the universe of discourse.

Let $\boldsymbol{X} = \{x_1,\ x_2,\ x_3,\ x_4,\ x_5\}$, and $\boldsymbol{A}_C = \{x_1,\ x_2,\ x_3\}$ and $\boldsymbol{B}_C = \{x_3,\ x_4\}$ be the crisp subsets of $\boldsymbol{X}$. Figure 7 shows a venn diagram representation of $\boldsymbol{X}$. We may represent $\boldsymbol{A}_C$ and $\boldsymbol{B}_C$ as the union of their elements:

$$A_C = x_1 + x_2 + x_3 \tag{4}$$

$$B_C = x_3 + x_4. \tag{5}$$

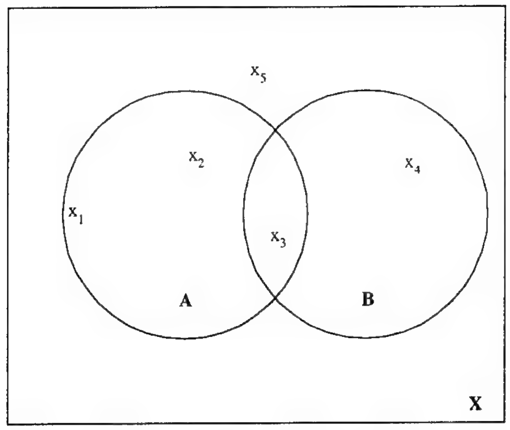


Figure 7 Venn Diagram of the Universe of Discourse, ***X***

Fuzzy set representation of ***A*** and ***B*** requires a membership function to grade the membership of all elements of ***X*** in ***A*** or ***B***. A suitable membership function is shown in Figure 8. All elements within the circle (see Figure 7) are graded as one; membership reduces radially in a linear manner and elements greater than two radii from the center of the circle are graded as having zero membership.

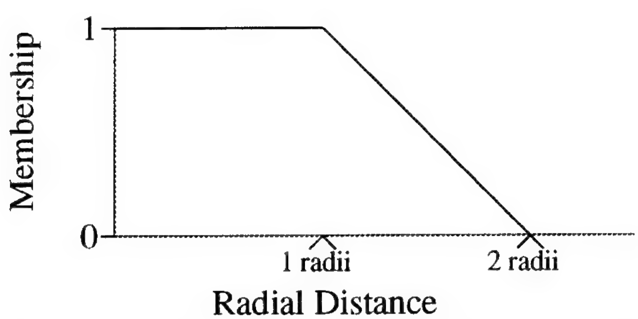


Figure 8 Membership Function for Subsets ***A*** and ***B***

In fuzzy logic, the general subset $\boldsymbol{A}$ may be expressed as

$$\boldsymbol{A} = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \cdots + \mu_A(x_n)/x_n; \tag{6}$$

thus, from Figure 8, we have

$$\boldsymbol{A} = 1/x_1 + 1/x_2 + 1/x_3 + 0.1/x_4 + 0.75/x_5, \tag{7}$$

and similarly,

$$\boldsymbol{B} = 0/x_1 + 0.5/x_2 + 1/x_3 + 1/x_4 + 0.5/x_5. \tag{8}$$

Elements x_3 and x_4 have full membership in $\boldsymbol{B}$, x_2 and x_5 have partial membership in $\boldsymbol{B}$, and x_1 has zero membership in $\boldsymbol{B}$ because its distance is greater than two radii from the center of $\boldsymbol{B}$.

The assignment of the membership function of a fuzzy set is subjective in nature and should be based on the potential application and properties desired. Although the assignment of a membership function is subjective, it cannot be assigned arbitrarily. [Ref. 8] Consider the universe of discourse of real numbers greater than one. Define the fuzzy subset

$$\boldsymbol{A} = \left\{ \frac{\mu_A(x)}{x} \right\}, \quad x \in \boldsymbol{X} \tag{9}$$

where $\boldsymbol{X}$ is a set of real numbers much larger than one. A possible membership function for $\boldsymbol{A}$ is

$$\mu_A(x) = \begin{cases} \dfrac{x-1}{x}, & \mathit{for}\ x \geq 1, \\ 0, & \mathit{otherwise}\ , \end{cases} \tag{10}$$

on the other hand, it would be incorrect to assign the membership function as

$$\mu_A(x) = \begin{cases} 0, & \mathit{for}\ x \leq 1, \\ e^{-(x-1)}, & \mathit{for}\ x > 1. \end{cases} \tag{11}$$

In (11) as x gets large the membership of x in $\boldsymbol{A}$ becomes smaller which is not the desired membership function.

B. BASIC OPERATIONS

The operations from crisp set theory are applicable to fuzzy set theory as well. Basic set theory operations include inclusion, intersection, union, complement and equality. Let us now examine how these operations are implemented in fuzzy set theory.

In classical set theory $\boldsymbol{A}$ is included in $\boldsymbol{B}$ if every element of $\boldsymbol{A}$ is an element of $\boldsymbol{B}$. This is written as $\boldsymbol{A} \subseteq \boldsymbol{B}$. In fuzzy set theory, $\boldsymbol{A}$ is included in $\boldsymbol{B}$ if and only if

$$\mu_A(x) \leq \mu_B(x), \quad x \in \boldsymbol{X}. \tag{12}$$

For example, suppose $\boldsymbol{X} = \{x_1, x_2, x_3, x_4\}$, $\boldsymbol{A} = 0.1/x_1 + 0.5/x_2 + 0/x_3 + 1/x_4$ and $\boldsymbol{B} = 0.1/x_1 + 0.6/x_2 + 0.3/x_3 + 1/x_4$. $\boldsymbol{A}$ is included in $\boldsymbol{B}$ since $\mu_A(x_1) \leq \mu_B(x_1)$, $\mu_A(x_2) \leq \mu_B(x_2)$, $\mu_A(x_3) \leq \mu_B(x_3)$, $\mu_A(x_4) \leq \mu_B(x_4)$ and $\mu_A(x_5) \leq \mu_B(x_5)$.

In classical set theory, $\boldsymbol{A}$ is the complement of $\boldsymbol{B}$ if $\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{X}$ and $\boldsymbol{A}\ \boldsymbol{B} = \emptyset$, where $\emptyset$ is the null set. In fuzzy set theory, $\boldsymbol{A}$ and $\boldsymbol{B}$ are complementary if and only if

$$\mu_A(x) = 1 - \mu_B(x), \quad x \in \boldsymbol{X}. \tag{13}$$

For example, if $\boldsymbol{B} = 0.1/x_1 + 0.6/x_2 + 0.3/x_3 + 1/x_4$, then $\boldsymbol{B}^C = 0.9/x_1 + 0.4/x_2 + 0.7/x_3 + 0/x_4$.

The intersection of fuzzy sets $\boldsymbol{A}$ and $\boldsymbol{B}$ is the largest fuzzy subset contained in both fuzzy subsets $\boldsymbol{A}$ and $\boldsymbol{B}$. The intersection of $\boldsymbol{A}$ and $\boldsymbol{B}$ is expressed as a membership function given by

$$\mu_{A \cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right), \quad x \in \boldsymbol{X}. \tag{14}$$

From Figure 7 and (7) and (8), the intersection of $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by

$$\mu_{A \cap B}(x) = 0.5/x_2 + 1/x_3 + 0.1/x_4 + 0.5/x_5. \tag{15}$$

The union of $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as the smallest fuzzy set containing both $\boldsymbol{A}$ and $\boldsymbol{B}$. The union of $\boldsymbol{A}$ and $\boldsymbol{B}$ is expressed as a membership function given by

$$\mu_{A\cup B}(x) = \mathbf{max}\,(\mu_A(x),\ \mu_B(x)),\quad x \in \boldsymbol{X}. \tag{16}$$

Using the same example as above, the union of $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by

$$\mu_{A\cup B}(x) = 1/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + 0.75/x_5. \tag{17}$$

The algebraic product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is characterized by the following membership function

$$\mu_{AB}(x) = \mu_A(x)\cdot\mu_B(x),\quad x \in \boldsymbol{X}, \tag{18}$$

and their algebraic sum is characterized by

$$\mu_{A\dotplus B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\cdot\mu_B(x),\quad x \in \boldsymbol{X}. \tag{19}$$

C. DESIGNING A FUZZY SYSTEM

Applications of fuzzy systems can be found in several areas including, pattern recognition, image processing and control systems. By far the largest use of fuzzy systems is in controls. In this section, the problem of balancing an inverted pendulum is presented as an example to design a fuzzy system. First we summarize the classical approach to the problem and then present the solution based on the fuzzy system approach. The inverted pendulum system is shown in Figure 9.

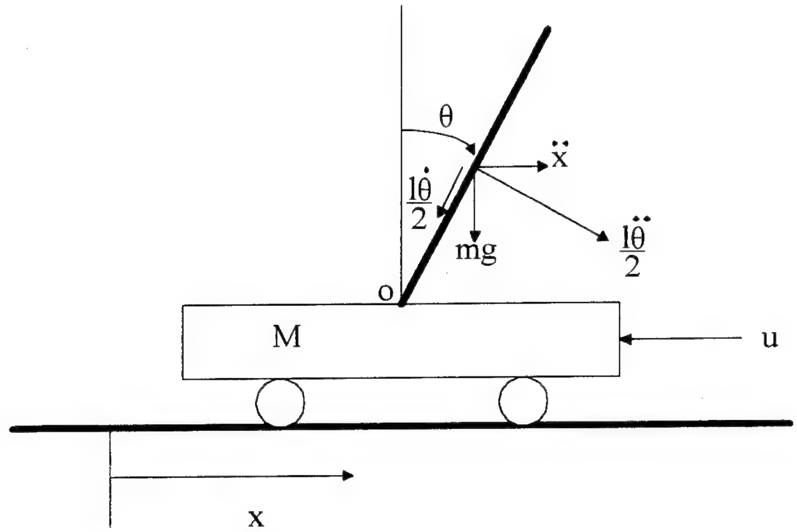


Figure 9 Inverted Pendulum

1. Classical Control Design Approach

The classical control design approach requires a mathematical description of the system. Once the mathematical model is developed, it is common to linearize it for the ease of analysis. The resulting solution to the linearized differential equations is in the neighborhood of the solution of the non-linear differential equation. Assuming that the system parameters remain within the linearization assumptions, the control solution is generally valid. These are the assumptions made to linearize the inverted pendulum problem: θ and $\dot{\theta}$ are small; M is large; and M»m. Consequently, $\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$ and $\dot{\theta}^2 \approx 0$. After linearization and much simplification the equations that describe the inverted pendulum problem are [Ref. 9]:

$$\ddot{x} = \frac{-1}{M+m}\theta - u \tag{20}$$

$$\ddot{\theta} = \theta + u \tag{21}$$

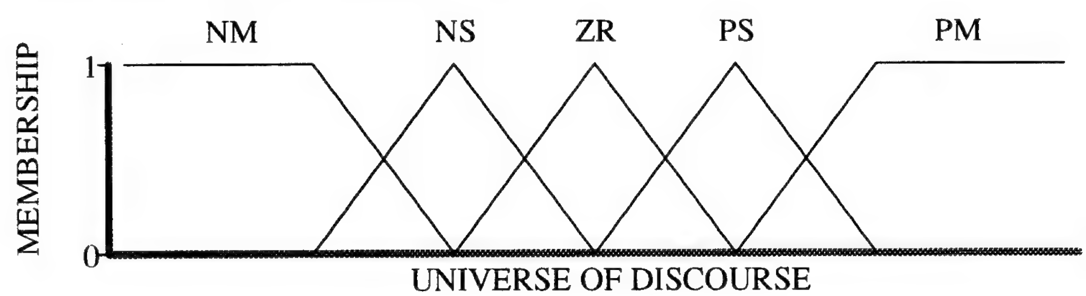
where $\ddot{x}$ is the cart's acceleration and M is the mass of the cart, m is the mass of the pendulum, θ is the pendulum's angular displacement from vertical, $\ddot{\theta}$ is the pendulum's angular acceleration, and u is the corrective force applied to the cart. The linear system described in (20) and (21) is known as a linear regulator. [Ref. 10] This type of system dynamically calculates a feedback gain matrix to minimize the output. The design is complete when the feedback gains are known.

2. Fuzzy Logic Design

There are four major steps involved in designing a fuzzy system:

1. Determine the universe of discourse of inputs and outputs.
2. Design membership functions.
3. Choose fuzzy rules to relate the inputs to the outputs.
4. Determine a defuzzier technique.

The inputs to the inverted pendulum controller are θ, $\dot{\theta}$, x, and $\dot{x}$. As previously discussed the selection of the membership functions is somewhat arbitrary but should be tailored to yield desired system response. The most common membership functions are triangular and trapezoidal in shape. The universe of discourse and membership functions for each of the inputs is shown in Figure 10.



-75 -60 -45 -30 -15 0 15 30 45 60 75 θ

-30 -20 -10 0 10 20 30 $\dot{\theta}$

-10 -5 0 5 10 x

-5 -2.5 0 2.5 5 $\dot{x}$

Figure 10 Input Membership Functions

The fuzzy subsets are defined as: negative medium (NM), negative small (NS), zero (ZR), positive small (PS) and positive medium (PM). The range for each input quantity is shown along the x-axis. The subsets NS, ZR and PS are smaller than the rest to provide for finer control when an input is near zero. Subsets NM and PM give coarse control. This assignment of fuzzy subsets makes engineering sense and will help to avoid overshoot when operating near the optimum solution.

The output of the fuzzy controller is the force (u) applied to the cart. The output universe of discourse is defined with its associated membership functions in Figure 11.

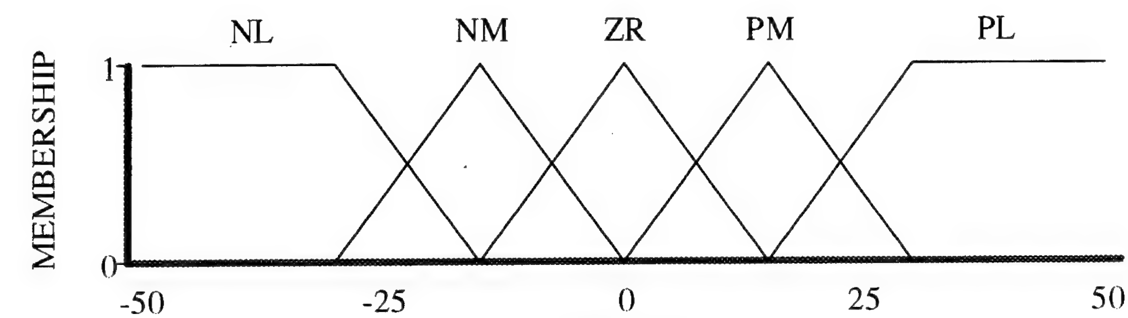


Figure 11 Output Membership Functions

The next step is to choose the fuzzy rules. Fuzzy *if-then rules* relate input subsets to output fuzzy subsets. Using each input separately will require up to 5^4 (625) rules. These rules are of the type: *if θ is NS and $\dot{\theta}$ is NZ and x is PS and $\dot{x}$ is NZ then u is PS.* A rule for most possible combination of the inputs is required. This obviously leads to too many rules to work with. A common technique to reduce the number of if-then rules is to combine inputs. For this problem a linear combination of the inputs may be used. Let $\bar{\theta} = a\theta + b\dot{\theta}$ and $\bar{x} = cx + d\dot{x}$ where a, b, c, d are scalars. The maximum number of rules is now 5^2. In practice many rules may be eliminated. [Ref. 11] A rule map may be used to describe the resulting if-then rules. The rule map relating the inputs $\bar{\theta}$ and $\bar{x}$ to the output u is shown in Figure 12.

$$\bar{\theta} = a\theta + b\dot{\theta}$$

$\bar{x} = cx + d\dot{x}$	NM	NS	ZR	PS	PM
PM	PL		PM		NL
PS		PM		NM	
ZR	PL	PM	ZR	NM	NL
NS		PM		NM	
NM	PL		NM		NL

Figure 12 Rule Map of Fuzzy If-Then Rules

In a fuzzy system all rules are applied to sum degree. The output is dependent on the type of defuzzier used. Figure 13 shows a typical fuzzy system. A defuzzier determines the output based on all of the inputs from the if-then rules. The most common defuzzier is the centroidal type, but minimum and maximum defuzziers are also used. Centroidal defuzziers yield continuous outputs whereas minimum and maximum types yield discrete outputs.

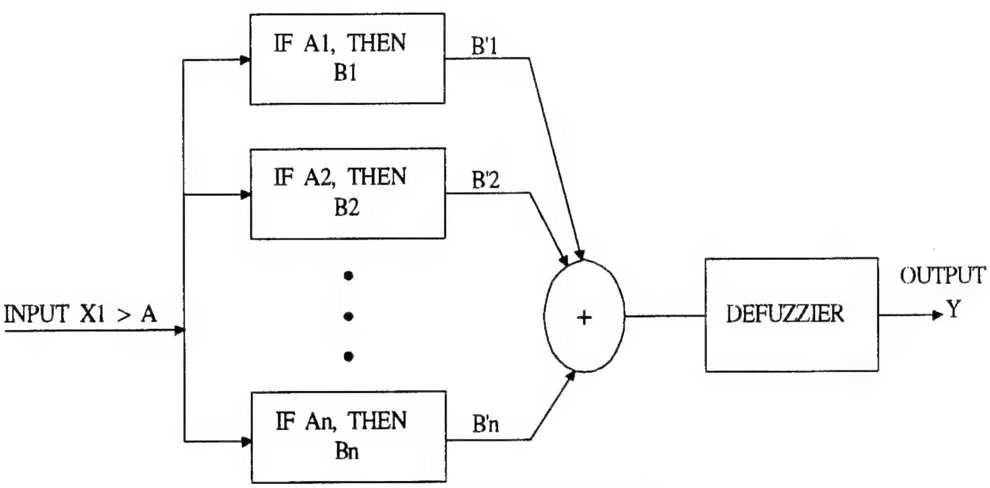


Figure 13 Fuzzy System

The output of a centroidal defuzzier is the centroid of the weighted membership functions. Consider the example shown in Figure 14. Let $\bar{\theta}$=2 and $\bar{x}$=-0.5 which represent a situation where the pendulum has a positive and increasing θ and the movement of the cart is in the negative x direction.

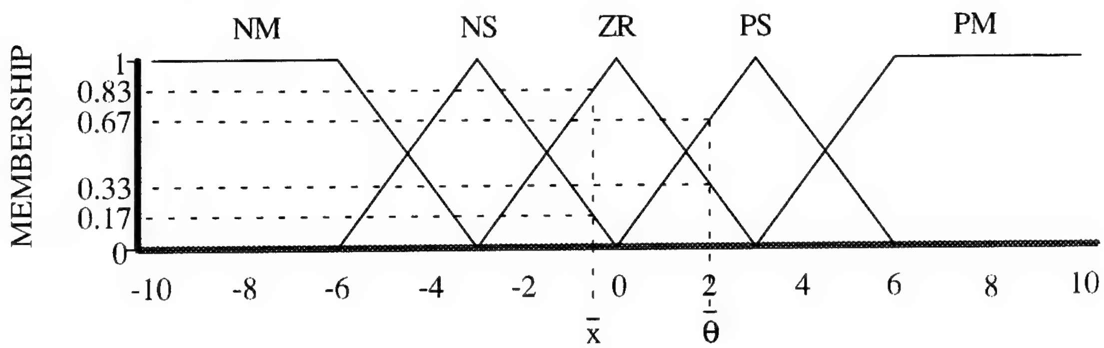


Figure 14 Inputs to Fuzzy System

Recall $\bar{\theta}$ is a linear combination of θ and $\dot{\theta}$, and $\bar{x}$ is the linear combination of x and $\dot{x}$. $\bar{\theta}$ has 0.33 membership in subset ZR and 0.67 membership in PS. $\bar{x}$ has 0.17 membership in NS and 0.83 in ZR. The rules affected by this input are found in the rule map. The membership of each output subset is determined by using the "min" operator of the input memberships. The "min" operator is used because the inputs are being combined with a logical "and." The rules that lead to non-zero values are given by

$$\begin{array}{lll} \text{IF } \bar{\theta} \text{ is ZR and } \bar{x} \text{ is ZR THEN U is ZR} & \xrightarrow{\min} & 0.33 \text{ ZR} \\ \text{IF } \bar{\theta} \text{ is PS and } \bar{x} \text{ is NS THEN U is NM} & \xrightarrow{\min} & 0.17 \text{ NM} \\ \text{IF } \bar{\theta} \text{ is PS and } \bar{x} \text{ is ZR THEN U is NM} & \xrightarrow{\min} & 0.67 \text{ NM} \end{array} \Bigg\rangle \xrightarrow{\max} 0.67 \text{ NM}.$$

When more than one rule results in the same output subset being selected, the "max" operator or logical "or" is used to determine the membership.

The outputs of the fuzzy if-then rules are applied to the centroidal defuzzier, as shown in Figure 15. The centroid of the weighted membership functions is the resultant force applied to the cart. The centroid of this area is -12.2; therefore, a force of -12.2 is applied

to the cart. This force will pull the cart in the positive x direction, which is required to correct the pendulum.

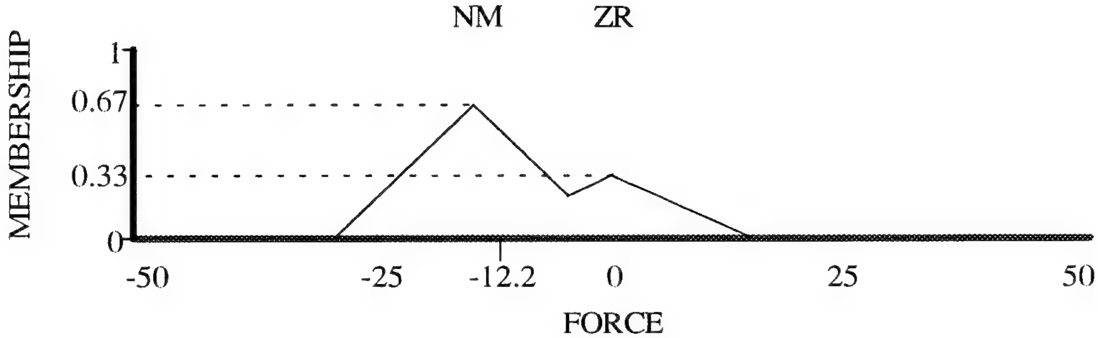


Figure 15 Centroidal Defuzzier Input and Output

3. Comparisons

The previous discussion shows the advantages and disadvantages of the two approaches. Fuzzy logic design does not require development of a mathematical model but requires experimentation to determine coefficient parameters (a, b, c, d), the fuzzy rules, and fine tuning of the membership functions to obtain the desired output. Classical design techniques are well established, and there are many design aids available to implement the system once the mathematical description of the problem is known. Assuming that the non-linearities and random disturbances remain within the assumptions of the mathematical model, a working controller may be implemented without additional experimentation. The choice of the design technique is dependent on the system. If a mathematical model contains many non-linearities, the fuzzy approach may be a better method whereas if development of the mathematical model is straightforward, the classical methods are probably a better design choice.

V. ALGORITHM

The goals of the proposed fusion algorithm are to remove redundant observations, incorporate differential GPS to improve radar position accuracy, automate historical track data, and ensure that the algorithm is computationally efficient. It is located in the sensor data processor and performs central level positional data fusion to fuse redundant observations. Radar track data from multiple remote sites and differential GPS from reporting vessels are processed for display and storage. The membership function approach from fuzzy set theory is used to provide the measure of correlation when associating observations and a fuzzy associative system is used to determine which observation pairs to group. A harbor simulation has been generated to test the performance of the algorithm. The various parts of the algorithm, details of the harbor simulation, and finally results of tests are discussed in the following sections.

The computer code for the algorithm was written in Matlab 4.0. Appendix B contains a listing of the algorithm functions. A listing of functions for simulation of the VTS system is provided in Appendix C.

A. INPUTS TO THE ALGORITHM

The inputs to the algorithm are radar track reports from the remote site radar processors and DGPS reports from reporting vessels. The radar reports and DGPS reports each contain position and kinematic information. Additionally, the differential GPS report contains identification information as well. Sample radar observations and DGPS reports used in the simulation are shown in Table 1. The origin of the data, update rates, and generation of these inputs are discussed in the following subsections.

1. Radar Data

The raw radar data from each scan is processed by the radar processor to develop the radar tracks. A target detection occurs when the magnitude of the echo return and the angle time on target (ATOT) exceed established thresholds. Track initiation occurs when M

detections occur in N scans; these criteria are established by the tracking algorithm in the remote site radar processor. This processing removes false detections and greatly reduces false tracks. This is important because the algorithm assumes each reported radar track to be a valid track.

Radar Data		DGPS Data	
2.72	x position	9.47	x position
13.84	y position	5.33	y position
273.06	course	298.04	course
9.26	speed (knots)	15	speed (knots)
90	time (epoch)	360	time (epoch)
0	maneuvering zone	A	identifier
BS	remote site	6	number of satellites
2	track number		

Table 1 Simulated Data

Typically, radar processors report the position of a target as the centroid of the of the ATOT and the range. The closer a target is to the radar, the larger the total angular extent of the target. Figure 16 illustrates how a vessel occupies a larger ATOT as the vessel approaches the radar. When a vessel is farther out, it has a smaller total angular extent, and the centroid of this ATOT fluctuates less from scan to scan than the larger ATOT of a vessel in close range. The effect of this is reduced position accuracy when a vessel is close to the radar. Between "close-in" and "far-away" the position accuracy of a vessel is aspect dependent. If a vessel's aspect presents a narrower target (e.g. vessel's bow or stern facing radar), its ATOT will be smaller than if the vessel were to present a full port or starboard aspect. These radar characteristics are used to model noise in the simulation program, which is used to evaluate the algorithm.

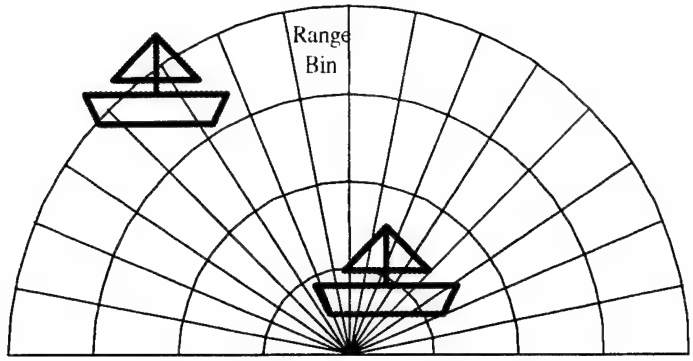


Figure 16 Radar Characteristics

After a target has been detected the tracking algorithm performs sensor level fusion to determine whether this detection is a new or false target, or an update to an existing track. The identifying characteristic of the radar track reports is the track number assigned to each radar report. The output of the radar processor is the new and updated position, kinematics and track number of each target being tracked within its search volume. Recall that the typical update rate of radar data processors used in the VTS system is 20 - 30 scans per minute.

Simulated radar data were generated by selecting the desired path and generating radar observations at three second intervals along that path. Each observation was then checked to see which radars could observe that position. In areas of overlapping radar coverage multiple observations were generated. The remote site identity and a track number were attached to each observation. Several track scenarios were generated: a single vessel in overlapping radar coverage area; one vessel passing another on a parallel course and in close proximity; crossing vessels; vessels maneuvering close to the radar; and the remote site radar processor updating a vessel with an incorrect track number.

Position noise was added to the radar data in accordance with the radar characteristics described earlier. Zero mean uniformly distributed noise with the standard deviation dependent on the observation's range from the reporting radar was added to each observation. Figure 17 shows the function used in modeling the position noise added to the radar observations. Zero mean uniformly distributed noise with a standard deviation of 10 degrees

was added to the course; the speed is corrupted with a zero mean uniform noise with a standard deviation of two knots.

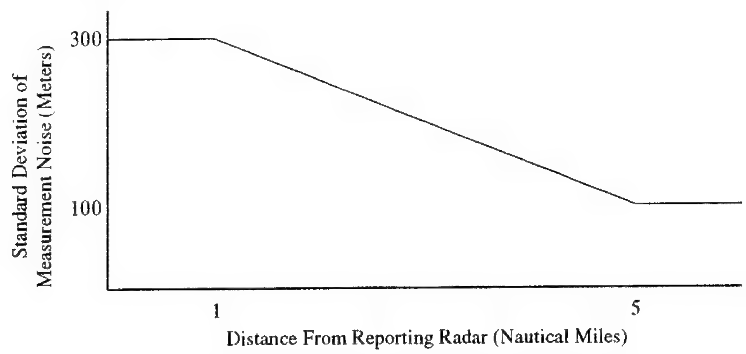


Figure 17 Additive Noise Model of Radar Characteristics

2. DGPS Data

The DGPS system consists of 24 satellites broadcasting continuous satellite position information. A receiver uses this information as well as time and distance information from each of the satellites in view to calculate its position. Four satellites are required for accurate position estimates. Position accuracy is dependent on the mode of operation. The USCG plans to provide differential GPS (DGPS) service for harbor and harbor approach phase areas of the VTS equipped ports. [Ref. 3] Differential GPS is an enhancement to the GPS system. DGPS is based upon accurate information of a fixed GPS reference station which is used to compute corrections to the GPS position reports. The corrections are then broadcast to authorized users who apply the corrections to their basic GPS position estimate. These corrections improve accuracy to less than 10 meters, compared to 100 meters without the corrections. The 1992 Federal Radionavigation Plan states that the USCG will provide the DGPS corrections for U.S. harbor and harbor approach areas by 1996.

The DGPS report from reporting vessels contains much more information than just position. These reports may contain the vessel's name, length, destination, cargo, whether military or civilian and the number of satellites being observed. The reports are broadcast to the vessel traffic control via the radio communications aboard the vessel. The update rate of these reports is unreliable and may be fewer than 10 reports per hour.

In the simulation the DGPS data were generated by offsetting the noise-free radar observations. DGPS range errors may be expressed as a zero mean Gaussian process with a specified accuracy. [Ref. 3] Zero mean Gaussian noise with a standard deviation of 10 meters was then added to the DGPS position information.

B. FUSION ALGORITHM

The proposed algorithm performs central level data fusion to remove redundant radar observations, utilizes DGPS reports to improve the accuracy of radar position reports and automates the track history of transiting vessels. Figure 18 is a flowchart of this algorithm. As can be seen in the flowchart, multiple remote sites provide radar track data; vessels equipped with GPS receivers provide DGPS reports to the algorithm. The following sections discuss the fusion algorithm using tables of results and additional discussion presented in Appendix A. Appendix A contains an example which presents a VTS traffic scenario based on the harbor geography at Governor's Island, New York and clearly illustrates the functioning of the fusion algorithm. The discussion here assumes only dual radar coverage of any position; however, the principles of the algorithm can be extended to any number of radars with overlapping regions of coverage.

1. Preprocessing

Data is collected for a period of t seconds. The selection of t is dependent on the desired update rate of the operator display and the typical number of vessels operating in the surveillance area. As the number of vessels increases, the computational requirements of the algorithm increases. For testing, data were collected for 9 and 15 second "windows." To simulate a missing radar report, radar observations were randomly eliminated. In the simulation, radar loss rates up to 30% were used.

The radar data and the DGPS data are then separated. The current DGPS correction matrix (Table 6, Appendix A) is used to correct the position of each radar observation. All corrected radar observations are stored in mass storage for use during DGPS data processing. The DGPS data are used to generate the correction matrix. The generation of the DGPS

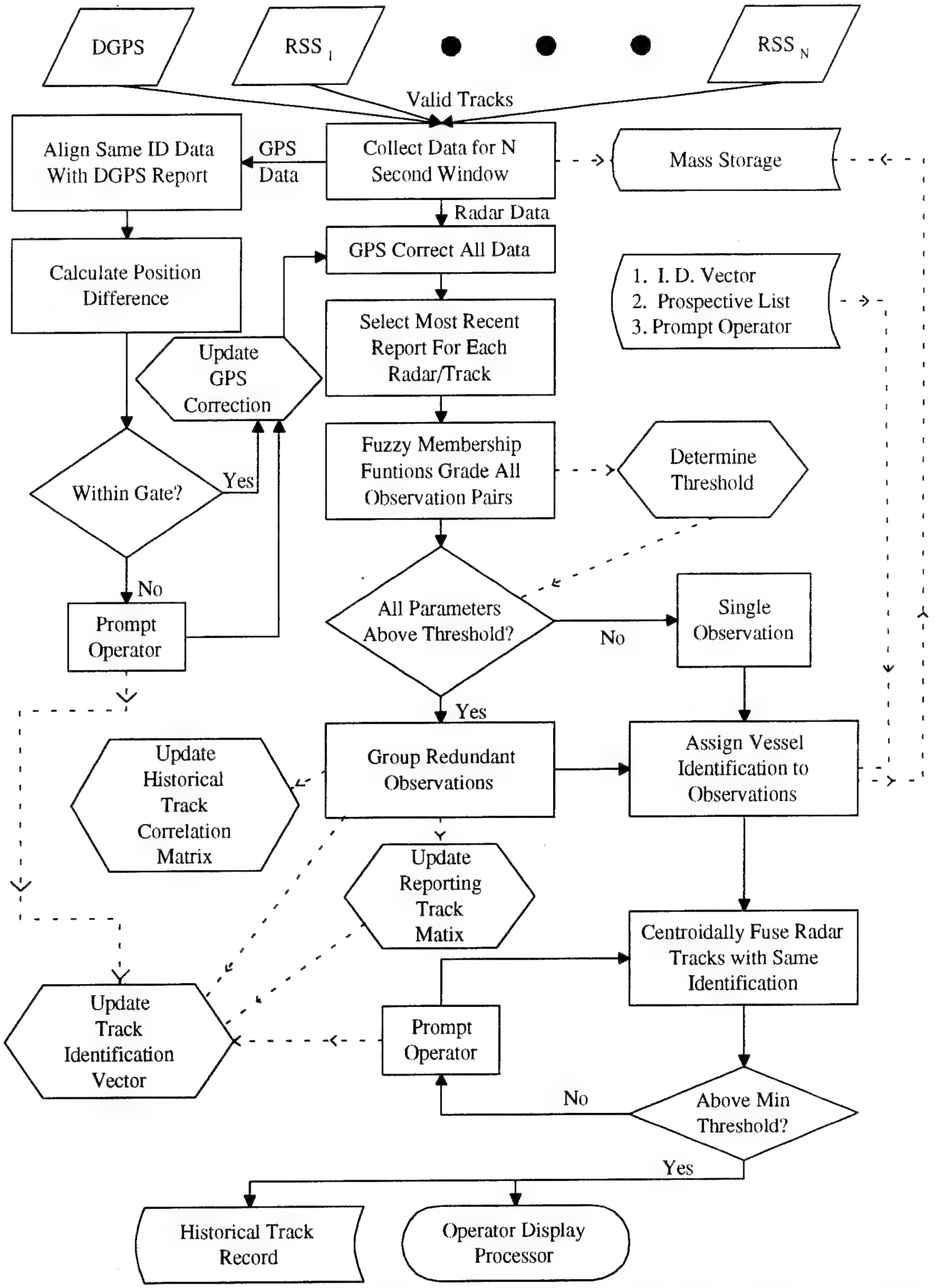


Figure 18 Flowchart of the Algorithm (The dashed lines indicate a subprocess, such as data retrieval/storage or update of a matrix.)

correction matrix will be discussed in part 5 of this section. Appendix A (Table 7) contains an example of a 15 second window of data and the DGPS correction matrix applied to the observations.

A reporting matrix (Table 9, Appendix A) is updated during each processing window. This matrix is used to erase identities from the identifier vector (Table 14, Appendix A) when a track fails to report. Each non-zero entry in the reporting matrix is decremented by one and then those tracks which have reported during this processing window have their value increased by the reporting weight. The reporting weight is generally set to one or two. When the reporting matrix entry of a track goes to zero, the identity assigned to that track in the identifier vector is erased. The identifier vector contains current identities for each track in the system.

2. Alignment

In the harbor environment vessel speed is greatly reduced. It is common for vessels to transit at speeds less than 15 knots. At 10 knots a vessel transits 15 meters in three seconds. Recall that the update rate of the radar processors in this system is between 20 and 30 per minute. Due to the update rate of the radar processors and the use of a collection window, there are now multiple reports of all vessels being tracked available for processing. The algorithm selects the most recent update for each track. If the loss rate is small, the most recent reports of each track will be "aligned" to within three to six seconds. This accuracy is within the association tolerances used when selecting observations for pairing during the association phase. From the example used in Appendix A (Tables 7 and 8), this method reduced the number of observations for processing from 21 to 8.

3. Association

The association phase seeks to associate similar radar observations. Fuzzy logic membership functions are used as a measure of correlation to compare observations to one another. Membership functions may be developed for any attribute of the data, such as

position, course, and speed. The membership functions used in the simulation program are shown in Figure 19.

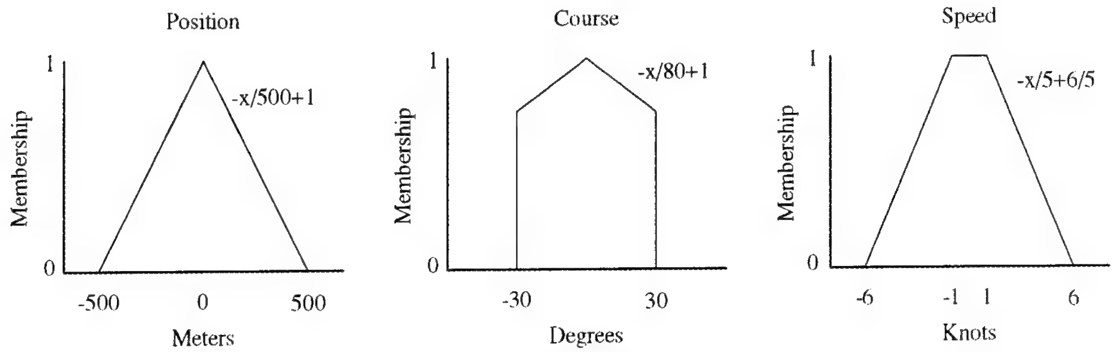


Figure 19 Membership Functions Used in Fuzzy Associative System

The design of the membership functions should consider the accuracy of the parameter it is being used for. For example if the output of the radar processor has accurate position estimates but poor course estimates, then the position membership function should be triangular and narrow, and the course should be trapezoidal with a flat region of values of one.

Prior to the evaluation of the observation pair by the fuzzy associative system, a minimum threshold for association of the pair must be established. A variable threshold is used in the algorithm due to the radar characteristics discussed previously. The threshold determines how "similar" the observations must be in order to be 'associated'. If a single threshold were established it would have to be set low enough to ensure association of vessels operating close to the radar. It is desirable to set the threshold for association as high as possible to ensure the observation pairs being associated are as similar as possible. Therefore, this algorithm uses a variable threshold dependent on the observation pair's distance from the reporting radars. Observations close to the radar have a lower threshold for association than those far away. A threshold is calculated for each observation pair based on the distance of the observation pair from their respective radars. The shortest distance among the pair is used for calculating the observation pair's *association threshold.* All association threshold values are stored in the variable association threshold matrix (Table 11, Appendix A). Figure 20 illustrates some of the variable threshold profiles used in the simulation.

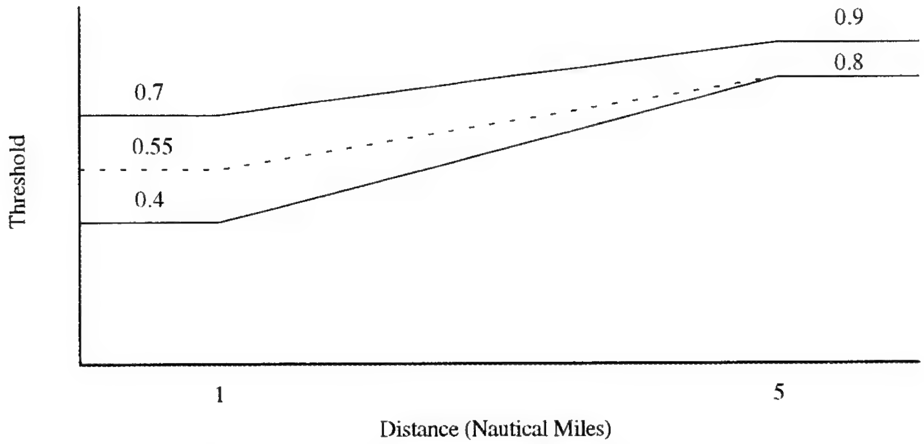


Figure 20 Variable Threshold Functions

A fuzzy associative system is used to determine if an observation pair represents the same vessel. The system compares all possible combinations of observation pairs. If there are n observations in a processing window, then this system must evaluate n^2-n observation pairs. The if-then rules used to relate the input observation pair to the output are depicted in Figure 21. These rules are based on the attributes available in the radar data. The output of the system is binary, i.e., whether or not the observation pair represents the same vessel. To reduce computation the defuzzier inspects the output of each rule (a membership value) sequentially to ensure that it is above the association threshold for that observation pair. If a membership value is below the association threshold, grading ceases, and the output of the defuzzier is set to zero. To obtain an output of one from the fuzzy system requires all four rules to have memberships above the association threshold for that observation pair. The association matrix (Table 12, Appendix A) contains the output of the fuzzy associative system for an entire processing window. This matrix indicates which observation pairs are being grouped or associated in this processing window.

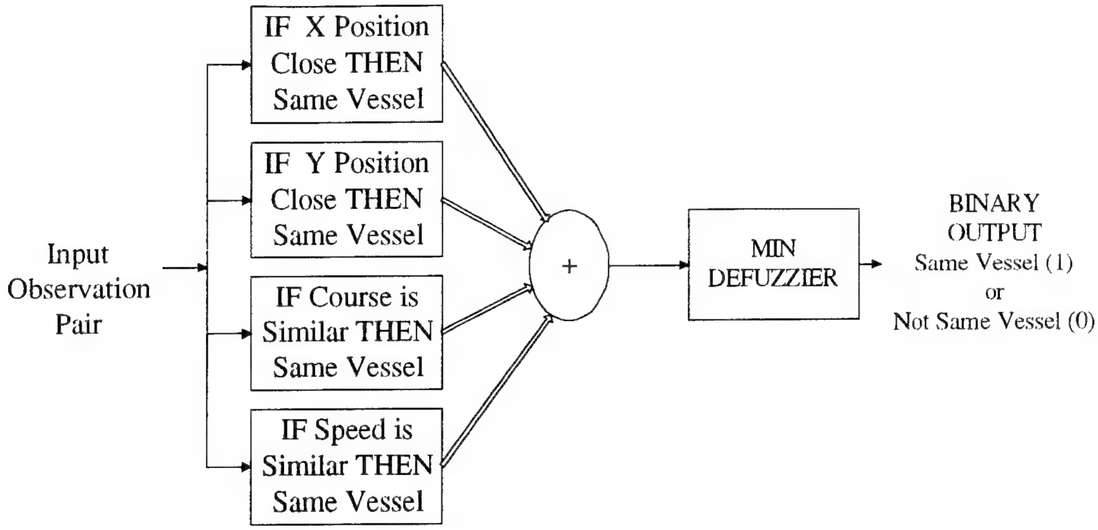


Figure 21 Fuzzy Associative System

In Chapter III we stated that if a track correlation history is maintained, tracks which were previously associated using more stringent criteria can be re-associated using less stringent criteria. This algorithm maintains a track correlation matrix (Table 13, Appendix A) to retain the association history; and establishes a reduced threshold for fusion of track pairs with a non-zero track correlation. The reduced threshold is different from, and lower than, the association threshold found in the variable association threshold matrix.

The association matrix is used to update the track correlation matrix. Initially all the track correlation entries are set to zero. During each processing window all non-zero entries in the track correlation matrix are reduced by one and then those tracks which have been associated are increased by the association weight. The association weight represents the 'value' of association during a processing window. The size of the weight determines how many subsequent processing windows the track pair will be allowed to fuse at the reduced threshold. The association weight is always greater than one. The effect of this is to allow fusion of track pairs which failed to be associated during a subsequent processing window, but would be associated with a reduced threshold. The track correlation entries are allowed to increase to an association maximum. The association maximum determines how many processing windows a track pair will be allowed to be fused at the reduced threshold without being associated.

The track correlation matrix is also used to 'hand-off' a vessel's identifier as the vessel transits the harbor. Prior to entering the estimation phase, all observations must have an identifier assigned to them because the estimation phase uses the identifiers to group the observations. The algorithm first checks the identification vector (Table 14, Appendix A) to retrieve the observation identity. If no identity is assigned to that track, which happens when a vessel is first observed by a radar, the algorithm checks the track correlation matrix to see if the track is correlated with another track. If the track is correlated with another track, the algorithm retrieves that track's identity and updates the identification vector with the identifier. If the track is not correlated with another track and has no identifier assigned in the identification matrix, then the prospective list is checked to determine if this observation is within a specified gate of any entries in this list. The prospective list is a list of vessels which have not entered the VTS area but have communicated their intention to do so. If an entry is found in the prospective list, then the identification vector is updated with the identity. If none of the three previous methods have provided an identity for the observation, then the operator is prompted for the identity.

4. Estimation

Estimation seeks to determine the state of each vessel being tracked. Observations with the same identifier are grouped together. Each group is processed individually. With only dual coverage, there should be a maximum of two observations in a group. If there are three of more observations with the same assigned identifier, the operator is prompted to correct the identifiers. If a group has only one member, no further processing is required for that group.

For groups with two members, the correlation matrix entry for the pair is checked. If the track pair's correlation is greater than zero, then the threshold value used for fusion is the reduced threshold. If the correlation matrix entry for the pair is zero, then the minimum value for fusion is the association threshold. If the minimum membership for the observation pair exceeds the minimum threshold allowed (association threshold or reduced threshold, as

appropriate), then the observation pair is fused. If the membership is lower than the minimum membership allowed, then the operator is prompted.

5. DGPS Data Processing

Valid DGPS reports from the reporting vessels are used to create a positional correction matrix for the radar data. The DGPS reports are not used for display because of their slow update rate (10 or less per hour). The vessel identification is used to match the DGPS data with a track. When a DGPS report is received, all tracks with the same identifier as the DGPS report are retrieved from mass storage. If no track is assigned the DGPS identifier, the positionally closest observation is selected, and the operator is prompted to determine if the observation should be used for the correction calculation. Alignment is performed on the lines of the method used with the radar data, i.e., selecting the temporally closest radar observation. The positional separation between the DGPS report and the radar observations is computed. The separation is checked to ensure that it is within a maximum gate. The gate size used in the simulation program is 500 meters. If the separation is less than the maximum gate size, then it is used to calculate the correction. The correction for each radar is the average of all previous corrections for that radar. If the separation is greater than the gate size, the operator is prompted. The operator can either accept the correction or disregard it.

C. RESULTS

Figure 22 illustrates the track scenarios used for testing; Figure 2 shows a detailed view of the overlapping radar coverage areas. Vessels A and G converge dead on the bow with a closest proximity of approach (CPA) of 27 meters (Figure 22 (b)). Vessel B transits from the Sandy Hook radar coverage area through an area of no radar coverage into the Bank Street radar coverage area (Figure 22 (e-f)). Vessels C and E have crossing tracks in an area of dual coverage with a CPA of 17 meters (Figure 22 (e)). Additionally, the first time the Governors Island radar tracks vessel E, the position of C and E is within 45 meters. Vessels

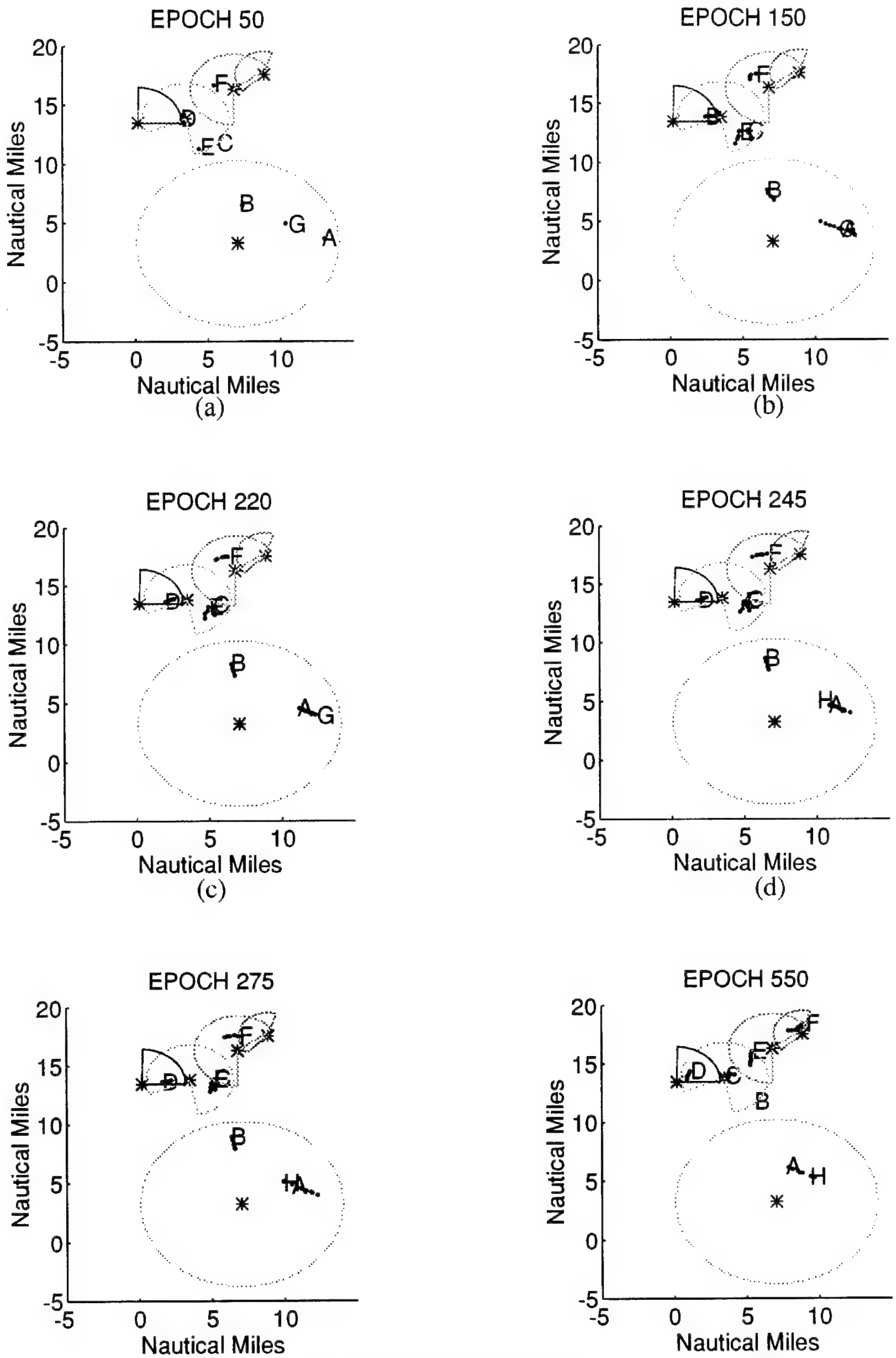


Figure 22 Track Scenarios

D and F operate close to the radars in the noisiest area for tracking. Vessels A and H operate on a parallel course with vessel A passing H with a CPA of 35 meters (Figure 22 (e-f)).

A maneuvering zone (MZ) parameter was added to evaluate the performance of the fusion algorithm. The algorithm was tested to evaluate the performance using all available information (e.g. position, course and speed) and using position information only, by setting the MZ flag appropriately. The algorithm was also tested to determine the effect of an error by a remote site radar processor. The remote site error was simulated by switching the track assignments of vessels C and E reported by the Bank Street site and A and G reported by the Sandy Hook site when the vessels were at CPA. Performance is judged on whether the algorithm needs to prompt the operator to determine an identifier for an observation; the goal is to have zero prompts.

1. Prompts

A prompt or interrupt of the operator is required when the algorithm is unable to determine the identity of an observation or when the algorithm has detected some type of anomaly. When prompted the operator provides information necessary for the algorithm to continue. In general prompts result from four conditions:

A: Track pair correlation matrix entry > 0, Membership < Reduced threshold,

B: Track pair correlation matrix entry = 0, Membership < Association threshold,

C: No identification assigned to the track, and

D: Three tracks with same identification.

Type A prompts occur when the observation pair has a membership below the reduced threshold for fusion. This can occur when the reduced threshold is set too high. The reduced threshold must be low enough to account for the worst case association, which occurs in the noisy region near the radars during high maneuvering and high loss rates. When the loss rate is low, the most recent report for each track is within a few seconds. As the loss rate increases the most recent report for each track become less temporally aligned. The misalignment must be accounted for by lowering the reduced threshold.

Type B prompts occur when the correlation of two previously correlated tracks goes to zero. When this occurs observations with the same assigned identity must have all membership values above the association threshold for the pair when attempting to fuse them; otherwise, the operator will be prompted. The main factor affecting this type prompt is the association weight and the association maximum. These parameters must be set high enough so that once correlation has been established, it is maintained through noisy periods or periods of operating close to another vessel.

Type C prompts occur when there is an observation with a track number which has no assigned identification. The factors which affect this type of prompt are the window size, reporting weight and loss rate. When a vessel first enters a region of multiple radar coverage, no historical correlation exists; thus the association threshold must be low enough to allow for association. If a redundant observation does not associate with its corresponding observation, then the operator is prompted to determine the vessel identification. Another factor affecting Type C prompts is an identification that has been erased due to a non-reporting track. The probability of no report can be kept small by increasing the window size or increasing the reporting weight. The reporting weight affects how long an identification is assigned to a track. After a vessel departs the radar coverage area, the radar processor must not assign a new vessel to that track for a period of time given by the window size and the reporting weight. If a new vessel is assigned the previous vessel's track before the identification has been erased, it will be assigned the previous vessel's identification.

In this simulation the are no areas of triple radar coverage; therefore an error occurs if three tracks are assigned the same identification (Type D). This can occur if the operator inputs the wrong identification on a previous prompt or if the algorithm associates a track with different tracks during separate processing windows; this usually occurs in an area of multiple coverage with vessels operating in close proximity. In dual coverage a track may only be correlated with one other track from a different sensor at any time.

Tables 2 and 3 show the results of tests conducted to determine the thresholds and weights necessary for no prompts to occur during the simulation program. The reduced threshold and the minimum of the variable threshold function (see Figur 20) are lower for the

individual site correction tests because this method must account for the error introduced by different corrections for interconnected sites.

Window	5	5	3	5	5
Association Weight	3	3	3	4	4
Association Maximum	6	6	6	8	8
Threshold Function Maximum	0.95	95	0.95	0.95	0.95
Threshold Function Minimum	0.65	0.60	0.60	0.65	0.70
Reduced Threshold	0.35	0.35	0.40	0.35	0.4
Loss Rate	0.2	0.2	0.3	0	0
Reporting Weight	1	1	2	1	1
Prompts	C, B	-	A, B	D	B, D
Frequency of Prompts	2, 2	-	2, 1	1	1, 1

Table 2 Individual Site Correction

Window	5	5	5	3
Association Weight	3	4	4	4
Association Maximum	6	8	8	8
Threshold Function Maximum	0.95	0.95	0.95	0.95
Threshold Function Minimum	0.70	0.65	0.65	0.60
Reduced Threshold	0.60	0.50	0.50	0.50
Loss Rate	0	0	0.3	0.3
Reporting Weight	1	1	1	2
Prompts	A, B	-	C, B	-
Frequency of Prompts	8, 1	-	1, 1	-

Table 3 Connected Sites Averaged

2. Trends

For a fixed window size as the loss rate increases, the association threshold and the reduced threshold must be lowered to ensure association of the most recent report of each track. Most of the prompts occur when vessels operate at close radar range, due to non-association of redundant tracks. Additionally, prompts tend to occur in groups due to non-association when a vessel first enters an area of dual radar coverage because the identity is not passed on when redundant tracks fail to associate. The minimum of the variable threshold function has the most effect on correcting these types of prompts. By reducing the minimum of the variable threshold function, the redundant observations become associated and future observations on these tracks may now be fused using the reduced threshold.

3. Normal Versus Degraded Mode

Normal mode of operation uses all available information and degraded mode uses position information only. The thresholds resulting in zero prompts for both modes were similar due to the construction of the course and speed membership functions and the amount of noise added to these parameters. The trends were the same for the two modes, the only exception being an occasional prompt due to the rapid change in course of vessel C from 350° to 270° while crossing with vessel E (Figure 22 (c-e)) during tests with high loss rates. Due to the narrow shape of the course membership function, the membership value would go to zero because the available observations were different by 30-50 degrees. At loss rates of 0-10% this error never occurred. Also changing the membership function to assign non-zero values to courses up to 50 degrees resulted in no prompts at loss rates of 30%.

The difference in the two modes is best described by discussing how each handles the scenario of the vessels crossing their paths. In normal mode the algorithm always associated correctly because the course information distinguishes the vessels. In degraded mode, a track is sometimes associated with up to three other tracks as the algorithm cannot distinguish between vessels until they are positionally separated in the distance established by the reduced threshold. In this simulation, the separation corresponding to the reduced threshold was 325 meters using individual site correction and 250 meters when an average of corrections from connected sites was used.

4. Errors

Tests were conducted to simulate the remote site radar processors erroneously switching the tracks which can occur when vessels are in close proximity. The track assignments for vessels A and G, and C and E were switched at CPA (Figure 22 (b-d)). Vessels A and G are being tracked in an area of single radar coverage. The algorithm is unable to detect the radar processor error under single radar coverage scenarios. The algorithm simply assigns the identification associated with that track.

In the area of dual coverage, tested by switching the track assignments of vessels C and E from the Bank Street site, the algorithm is able to detect this error in both modes.

Using all parameters the algorithm catches the error on the first report because the course memberships are below the reduced threshold of the observations assigned the same identifier. The operator is prompted, and the identification vector is updated. This results in no errors in the track history. In the position only mode the algorithm does not recognize that an error has occurred until the vessels have separated beyond the distance corresponding to the reduced threshold. Assuming the reduced threshold is 0.35, the vessels must separate by 325 meters before the error is detected. In this case it resulted in approximately 45 seconds of incorrect data being stored in the track history of the two vessels.

5. Computational Expense

Tests were conducted to determine if the Kalman filter method of alignment significantly improves the DGPS correction applied to the radar data. The results are shown in Table 4. The data with 0% loss indicate the correction required due to the noise added to the radar and DGPS reports. The error is determined by using the 0% loss corrections as the benchmark. The results show that the added expense of the Kalman filter is not justified. The data obtained for the case of averaging 15 reports show an error of 3.1 meters using the Kalman filter for alignment and 5.2 meters using the temporally closest method for alignment. The accuracy of the temporally closest method is considered adequate for this application.

	Sandy Hook	Connected Sites
0% Loss (No Alignment)		
x correction	-40.46	4.12
y correction	-9.32	59.74
reports averaged	6	15
Kalman Filter Alignment 30% Loss		
x correction	-39.38	1.33
y correction	-16.05	58.44
error	6.8	3.1
Temporally Closest Alignment 30% Loss		
x correction	-40.45	9.02
y correction	-8.4	57.96
error	0.9	5.2

Table 4 Comparison of Kalman Filter Alignment and Temporally Closest Alignment (all values are in meters)

This chapter has presented a description of the inputs to the fusion algorithm and the various steps of the algorithm. The discussion of the inputs is necessary to understand why a variable threshold is used during association of the redundant observations. The algorithm description follows the steps of positional fusion described in Chapter III. The algorithm achieves computational efficiency by collecting data for a time period t, and then selecting the most recent report of each track. Association of the aligned data is performed by a fuzzy associative system. The output of the fuzzy system is used to update a track correlation matrix. This matrix automates the track history and allows a reduced threshold for fusion of tracks previously associated. Tests of the algorithm indicate that with parameters of the algorithm adjusted correctly, the algorithm achieves the desired goals of track automation, fusion of redundant observations, incorporation of DGPS inputs, and computational efficiency.

VI. CONCLUSIONS

This thesis presented an overview of the current VTS systems, describing the need for improvements to remove redundant vessel displays and to automate the track history. Additionally the algorithm is required to incorporate DGPS reports to improve the accuracy of the radar position estimates. Computational efficiency is also important. The proposed algorithm achieves these goals by using a fuzzy system to associate and correlate similar observations.

The advantage of using a fuzzy associative system is the ease with which data attributes may be incorporated into the system. As more becomes known of the operating characteristics of the Telephonics processor, the membership functions of the fuzzy system may easily be adjusted to properly reflect the accuracy of the data attributes. Another advantage of using this method is its computational efficiency. By inspecting the output of the if-then rules sequentially, a decision may be made by the system with minimum computation.

Scenarios tested included vessels crossing paths and approaching within 17 meters. The fusion algorithm is able to distinguish and properly fuse the redundant observations due to the additional data attributes of course and speed. The algorithm was also tested using position information only. In this mode the algorithm cannot distinguish vessels until they are separated by a distance corresponding to the reduced threshold.

A major limitation of the algorithm is its error checking capability. The algorithm is able to detect an error by a remote site in an area of multiple radar coverage but unable to detect an error in single radar coverage. The algorithm does not check the data from the remote site, instead it assumes all observations to be valid.

To improve the error checking capability, a simple gate check comparing the current update of a track with the previous state could be implemented. The size of the gate is dependent on the processing widow size and the speed of the vessel being tracked. This adds considerable computational expense to the algorithm but could be performed only during

times when the radar processor is most likely to make an error, such as when a vessel is operating in close proximity to another vessel.

Test results indicate that the DGPS correction process directly affects the algorithm parameters required to achieve zero prompts of the operator. This is an area where the algorithm requires improvement. It is desirable to have individual site correction but until there are enough DGPS reports for an individual site to average, a weighted average technique should be implemented. The goal is to avoid large differences in the correction applied to overlapping sites. Another area where the algorithm needs improvement is the size of the algorithm variables. As written, the algorithm uses fixed size variables. This is acceptable for the simulation program where the remote sites were restricted to tracking three vessels, but in the VTS system there are up to 18 remote sites, each with the capacity to track up to 60 vessels. To accommodate this, the algorithm should be written to have variable size parameters dependent on the number of vessels being tracked.

The Coast Guard is considering removing the Database Processor or the Sensor Data Processor from the VTS system. Therefore the fusion algorithm could be used to perform the alarm processing currently performed in the Database Processor by fusing the vessel position information with *a priori* information of obstacles, geography and other vessel positions.

The GPS is a relatively new system. Currently many civilian vessel are not equipped with GPS receivers. As GPS becomes more widely used or perhaps required, these inputs should be used for tracking and not just for improved radar position accuracy. This will require an improvement in the update rate of the reports.

APPENDIX A: ALGORITHM EXAMPLE

An example that supports the discussion in Chapter V, Section B is presented here. The scenario considered consists of six vessels operating in the VTS surveillance area. Two of the six vessels are operating in areas of dual radar coverage. This appendix contains the algorithm matrices of a single processing window. The following settings were used for this trial.

Reporting Window (sec.)	15
Association Weight	4
Association Maximum	8
Threshold Function Maximum	0.9
Threshold Function Minimum	0.6
Reduced Threshold for Fusion	0.5
Loss Rate	0.3
Reporting Weight	2
Mode	0

Table 5 Algorithm Parameters

The following abbreviations of the remote sites are used in this appendix: Sandy Hook (sh), Mariners Harbor (mh), Bank Street (bs), Governors Island (gi), and Brooklyn Naval Yard (bny).

A. PREPROCESSING

The radar observations and DGPS reports are collected for 15 seconds. This reporting window contains one DGPS report and twenty one radar observations of the six vessels. The current DGPS correction is applied to each radar observation. Table 6 shows the current DGPS correction matrix, and Table 7 shows the corrected radar observations of this reporting window.

	Sandy Hook	Connected Sites (mh, bs, gi, bny)
x correction	-0.0377 NM (-69.82 meters)	-0.0172 NM (-31.77 meters)
y correction	0.0032 NM (5.93 meters)	0.0331 NM (61.39 meters)

Table 6 Current Correction Matrix

1	2	3	4	5	6	7	obs #
10.8500	6.3931	5.2266	10.8161	6.3905	1.833	4.9955	x position
4.6438	8.5990	13.5589	4.6620	8.5933	13.6819	13.5410	y position
306.51	001.21	342.38	302.49	358.79	258.65	25.86	course
14.24	13.11	10.53	13.02	14.36	6.85	14.36	speed
236	236	236	237	237	237	237	epoch
0	0	0	0	0	0	0	MZ
sh	sh	gi	sh	sh	mh	gi	remote site
1	2	3	1	1	1	2	track
8	9	10	11	12	13	14	obs #
5.1860	10.8552	6.4134	5.1982	6.5206	5.2018	1.8691	x position
13.6170	4.7156	8.6147	13.5961	17.5691	13.6445	13.7294	y position
341.73	291.09	344.37	345.07	66.49	343.20	266.57	course
11.41	13.19	13.06	12.23	11.42	11.58	7.93	speed
237	238	238	238	238	238	239	epoch
0	0	0	0	0	0	0	MZ
gi	sh	sh	bs	gi	gi	mh	remote site
3	1	2	3	1	3	1	track
15	16	17	18	19	20	21	obs #
1.7317	4.9401	5.2035	6.6448	4.9890	5.2194	6.5870	x position
13.6973	13.4899	13.6495	17.6093	13.4733	13.5521	17.6467	y position
272.75	14.07	005.45	91.33	14.44	358.62	89.02	course
8.96	12.61	12.81	11.94	12.96	10.65	11.73	speed
239	239	239	239	240	240	240	epoch
0	0	0	0	0	0	0	MZ
bs	bs	bs	gi	bs	bs	gi	remote site
1	2	3	1	2	3	1	track

Table 7 DGPS Corrected Observations in 15 Second Window

B. ALIGNMENT

The most recent report of each track is selected for further processing as shown in Table 8.

1	2	3	4	obs #
10.8552	6.4134	5.2194	1.8691	x position
4.7156	8.6147	13.5521	13.7294	y position
291.09	344.37	358.62	266.57	course
13.19	13.06	10.65	7.93	speed
238	238	240	239	epoch
0	0	0	0	MZ
sh	sh	bs	mh	remote site
1	2	3	1	track
5	**6**	**7**	**8**	**obs #**
4.9890	5.2018	6.5870	1.7317	x position
13.4733	13.6445	17.6467	13.6973	y position
14.44	343.20	89.02	272.75	course
12.96	11.58	11.73	8.96	speed
240	238	240	239	epoch
0	0	0	0	MZ
bs	gi	gi	bs	remote site
2	3	1	1	track

Table 8 Temporally Closest Aligned Observations

The reporting track matrix is updated as shown in Table 9. This matrix is used to erase identifiers from the identification vector. When a track goes to zero in the reporting track matrix that track's identifier is erased. In this case all of the current tracks have reported during this processing window and are increased by the reporting weight of two and then reduced to the maximum value of two.

	Remote Sites				
Track #	sh	mh	bs	gi	bny
1	2	2	2	2	0
2	2	0	2	0	0
3	0	0	2	2	0

Table 9 Reporting Track Matrix

C. ASSOCIATION

Each observation pair is graded to determine the membership of the observations as compared to one another. The diagonal represents the membership of each observation graded against itself and is not computed. The membership grading stops once a parameter is below the appropriate association threshold, saving unnecessary calculations. The membership of each observation pair is shown in Table 10, and the variable association threshold matrix is shown in Table 11. The values for observations three and six are shown in bold as these observations have all values above their association threshold.

obs #	1	2	3	4	5	6	7	8
1	- - - -	0 - - -	0 - - -	0 - - -	0 - - -	0 - - -	0 - - -	0 - - -
2	0 - - -	- - - -	0 - - -	0 - - -	0 - - -	0 - - -	0.3567 - - -	0 - - -
3	0 - - -	0 - - -	- - - -	0 - - -	0.1467 - - -	**0.9351** **0.6576** **0.8073** **1**	0 - - -	0 - - -
4	0 - - -	0 - - -	0 - - -	- - - -	0 - - -	0 - - -	0 - - -	0.4909 - - -
5	0 - - -	0 - - -	0.1467 - - -	0 - - -	- - - -	0.2117 - - -	0 - - -	0 - - -
6	0 - - -	0 - - -	**0.9351** **0.6576** **0.9323** **1**	0 - - -	0.2117 - - -	- - - -	0 - - -	0 - - -
7	0 - - -	0.3567 - - -	0 - - -	0 - - -	0 - - -	0 - - -	- - - -	0 - - -
8	0 - - -	0 - - -	0 - - -	0.4909 - - -	0 - - -	0 - - -	0 - - -	- - - -

Table 10 Membership of Aligned Observations

obs #	1	2	3	4	5	6	7	8
1	0.8296	0.9	0.6555	0.6515	0.6396	0.7575	0.6274	0.6576
2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
3	0.6555	0.9	0.6555	0.6515	0.6396	**0.6555**	0.6274	0.6555
4	0.6515	0.9	0.6515	0.6515	0.6396	0.6515	0.6274	0.6515
5	0.6396	0.9	0.6396	0.6396	0.6396	0.6396	0.6274	0.6396
6	0.7575	0.9	**0.6555**	0.6515	0.6396	0.7575	0.6274	0.6576
7	0.6274	0.9	0.6274	0.6274	0.6274	0.6274	0.6274	0.6274
8	0.6576	0.9	0.6555	0.6515	0.6396	0.6576	0.6274	0.6576

Table 11 Variable Association Threshold Matrix of Aligned Observations

Observations three and six have all membership values above their association threshold; therefore those observations are associated. The other observation pairs do not have all parameters above their respective association threshold and are not associated with any other track during this processing window. The association matrix in Table 12 shows the associations for this processing window; observations three and six are associated.

obs #	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	**1**	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table 12 Association Matrix

The association matrix is used to update the track correlation matrix. This matrix is used to pass track identifiers from one track to another. Observation three is the third track

of the Bank Street site, and observation six is the third track of the Governors Island site. The track correlation matrix for Bank Street track 3 (bs3) and Governors Island track 3 (gi3) is increased by the association weight, and all other previously correlated tracks are decremented by one. The previous value for this track correlation was eight. After increasing by four, it is reduced to the association maximum of eight. The updated track correlation matrix shown in Table 13 shows that the correlation of mh1 and bs1 is seven. The previous value of this pair was eight. This track pair's correlation has been reduced due to non-association of these tracks.

	sh1	sh2	sh3	mh1	mh2	mh3	bs1	bs2	bs3	gi1	gi2	gi3	bny1	bny2	bny3
sh1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sh2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sh3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mh1	0	0	0	0	0	0	**7**	0	0	0	0	0	0	0	0
mh2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mh3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bs1	0	0	0	**7**	0	0	0	0	0	0	0	0	0	0	0
bs2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bs3	0	0	0	0	0	0	0	0	0	0	0	**8**	0	0	0
gi1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gi2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gi3	0	0	0	0	0	0	0	0	**8**	0	0	0	0	0	0
bny1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bny2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bny3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13 Track Correlation Matrix

The current identifier vector is shown in Table 14. This vector contains the current identifier assigned to each track. This vector is updated automatically when track pairs are correlated or when a prospective list entry is selected. It is updated manually when the operator is prompted for an identity of an observation.

Remote Site	sh	sh	sh	mh	mh	mh	bs	bs	bs	gi	gi	gi	bny	bny	bny
track	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
identifier	A	B	-	D	-	-	D	E	C	F	-	C	-	-	-

Table 14 Track Identification Vector

Identifiers are assigned first from the identification vector, then by checking if a track is correlated with another track in the track correlation matrix. Next the prospective list is checked. If the previous three methods do not provide an identity, then the operator is prompted. In this case identifications are available for all of the observations and are assigned as shown in Table 15. The observations for vessels C and D are shown in bold because these vessels have multiple observations.

1	2	3	4	obs #
10.8552	6.4134	5.2194	1.8691	x position
4.7156	8.6147	13.5521	13.7294	y position
291.09	344.37	358.62	266.57	course
13.19	13.06	10.65	7.93	speed
238	238	240	239	epoch
0	0	0	0	MZ
sh	sh	bs	mh	remote site
1	2	3	1	track
A	B	**C**	**D**	identifier
5	6	7	8	obs #
4.9890	5.2018	6.5870	1.7317	x position
13.4733	13.6445	17.6467	13.6973	y position
14.44	343.20	89.02	272.75	course
12.96	11.58	11.73	8.96	speed
240	238	240	239	epoch
0	0	0	0	MZ
bs	gi	gi	bs	remote site
2	3	1	1	track
E	**C**	F	**D**	identifier

Table 15 Temporally Closest Aligned Observations

D. ESTIMATION

The observations with the same identifier are grouped together for estimation. Groups with a single observation correspond to observations in single radar coverage areas

(A,B, E, F) and do not require further processing. The first pair grouped is those with the identifier C. The track correlation for this pair (gi3 and bs3) is greater than zero (8), so fusion is allowed at the reduced threshold of 0.5. The membership values of the grouped observations are then computed, as shown in Table 16. All graded parameters for this pair are greater than the reduced threshold, so the centroid of each parameter is computed. The fused observation is used for display and historical track data.

Observation Pair		Memberships	Fused Observation
5.2194	5.2018	0.9351	5.2106
13.5521	13.6445	0.6576	13.5983
358.62	343.20	0.8073	350.91
10.65	11.58	1	11.12
240	238		239
0	0		0
bs	gi		bs
3	3		3
C	C		C

Table 16 Centroidal Fusion of Same Identifier Observations

The next observation pair for fusion processing is those with the identifier D. The track correlation for this pair (mh1 and bs1) is greater than zero (7), so fusion is allowed at the reduced threshold of 0.5 as well. The membership values of the grouped observations is then computed. The x position membership is below 0.5; therefore the operator is prompted to determine if these observations should be fused. The operator is shown the observation pair and asked whether or not to fuse them. The operator can assign new identifiers at this prompt if desired. In this case, say, the operator chooses to fuse the observations; the fused observation is shown in Table 17.

Observation Pair		Memberships	Fused Observation
1.8691	1.7317	0.4909	1.8004
13.7294	13.6973	0.8812	13.7134
266.57	272.75	0.9227	269.66
7.93	8.96	0.9937	8.45
239	239		239
0	0		0
mh	bs		mh
1	1		1
D	D		D

Table 17 Centroidal Fusion of Same Identifier Observations

E. DGPS DATA PROCESSING

In this processing window there is a single DGPS report. The algorithm searches the mass storage for tracks with the same identifier as the DGPS identifier and selects the temporally closest observation for the correction calculation. The DGPS report is valid since it has been generated from four or more satellites. The x and y corrections are calculated using the DGPS position as the reference. The DGPS report and the temporally closest radar observation with the same identifier is given in Table 18.

DGPS Report		Radar Obs		Correction
10.7936	x position	10.8552	x position	-114.177 meters
4.6967	y position	4.7156	y position	-34.862 meters
294.70	course	291.09	course	
14.83	speed	13.19	speed	
240	epoch	238	epoch	
A	identifier	0	MZ	
6	# of satellites	sh	remote site	
		1	track	
		A	identifier	

Table 18 DGPS Report and Temporally Aligned Same Identifier Radar Observation

The current correction for the Sandy Hook site is averaged together with the previous Sandy Hook corrections. The updated DGPS correction matrix is shown in Table 19.

	Sandy Hook	Connected Sites (mh, bs, gi, bny)
x correction	-0.0437 NM (-80.85 meters)	-0.0172 NM (-31.77 meters)
y correction	-0.0022 NM (-4.16 meters)	0.0331 NM (61.39 meters)

Table 19 Updated Type II DGPS Correction Matrix

APPENDIX B: ALGORITHM CODE

```
function ass_mat = assoc(mems, thresh_mat)
% function grp_vec = assoc(mems, thresh_mat)
% This function generates the association matrix based on the current membership values and the
% minimum threshold provided by thresh_mat
% INPUTS: mems - membership values of current observations
%        thresh_mat - minimum allowable threshold for observations (Variable threshold)
% Written by LT T. Ruthenberg USN
[r,col]=size(mems);
ass_mat=zeros(col,col);
nn=0;
for p=1:4:r-3
   nn=nn+1;
   for n=nn:col
         threshold=thresh_mat(n,ceil(p/4));
         if mems(p:p+3,n) > threshold
            ass_mat(ceil(p/4),n)=1;
         end
   end
end

function obs1=correct(obs, gps_correction)
% function obs1=correct(obs, gps_correction)
% This function corrects the current observations based on dgps input
% INPUTS:        obs - all observations within reporting window
%       gps_correction - (x,y) current correction for each radar
% Written by LT T. Ruthenberg, USN
[r,c]=size(obs);
obs1=obs;
for n=1:c
   obs1(1:2,n)=obs(1:2,n)+gps_correction(:,obs(7,n));
end

function track = fcreate(trak_from, trak_to)
% function track = fcreate(trak_from, trak_to)
% Function is called by fillup.m. Purpose is to create position reports in 3 second
% intervals from the time of the trak_from report to the time of the trak_to report
% INPUTS: trak_from - single observation from the track history preceding the trak_to observation
% Written by LT T. Ruthenberg, USN
num=trak_to(5,1) - trak_from(5,1);
if num==1
   track=[trak_from trak_to];
else
   delx=trak_to(1,1) - trak_from(1,1);
   dely=trak_to(2,1) - trak_from(2,1);
   crs=course(delx, dely);
   dist=sqrt(delx^2 + dely^2);
   incr_dist = dist/num;
```

```
   spd=(trak_from(4,1) +trak_to(4,1))/2;
   track=trak_from;
   for n=2:num
    track(1,n)=track(1,n-1)+delx/num;
    track(2,n)=track(2,n-1)+dely/num;
    track(3,n)=crs;
    track(4,n)=spd;
    track(5,n)=track(5,n-1)+1;
    track(6,n)=track(6,n-1);
    track(7,n)=track(7,n-1);
    track(8,n)=track(8,n-1);
   end
end
track=[track trak_to];

function full_trak = fillup(k_track)
% function full_trak = fillup(k_track)
% This function called by gpskalm.m to fill in the missing observations of the track history prior to sending
% to the Kalman filter, because the Kalman filter (as written) expects an observation every 3 seconds
% INPUTS: k_track - last 20 or so observations of a track history
% Written by LT T. Ruthenberg, USN
[r,c]=size(k_track);
full_trak=k_track(:,1);
for n=2:c
   if k_track(5,n-1)+1 < k_track(5,n)
          temp=fcreate(k_track(:, n-1), k_track(:,n));
          [row,col]=size(temp);
          full_trak=[full_trak temp(:, 2:col)];
   else
          full_trak=[full_trak k_track(:,n)];
   end
end

function [correction, rx_corr, nams] = gpsclos(epoch, track_hist, rx_corr, window, nams)
% function [correction, rx_corr] = gpsclos(epoch, track_hist, rx_corr, window)
% This function updates the correction made to each radar.  The radar report closest in time to the gps
% reporting time is used to determine the correction.
% INPUTS: epoch - current epoch in nysim
%       track_hist - history of each track
%       rx_corr - (x,y) all previous corrections for each radar
%       window - size of reporting window
% Written by LT T. Ruthenberg
load gps1  % contains v_gps generated in nydata1.m
load radarll
radar_radius=[7 3 3 3 2];
current_gps=[];
correction=zeros(2,5);
typ=2;
for n=1:length(v_gps)
   if epoch-window+1 <= v_gps(5,n) & epoch >= v_gps(5,n)
          current_gps=[current_gps v_gps(:,n)];
```

```
  end
end
                          % search track history for same identifier track
[r,c]=size(current_gps);
[rh,ch]=size(track_hist);
for n=1:c
  if current_gps(8,n) > 3 % 4 satellites req'd for accurate GPS report
        tracks=[];
        for z=10:10:rh
          row_time=z-5;
          temp=max(find(track_hist(row_time,:) > 0));
          col=max(find(track_hist(row_time,1:temp)<=current_gps(5,n))); % temporal alignment
          if col > 1 & abs(current_gps(5,n)-track_hist(row_time,col)) < 10
                tracks=[tracks track_hist(z-9:z,col)];
          end
        end
                          % search the temporally aligned tracks vector for same name vessel
        col=find(current_gps(7,n)==tracks(9,:));
        if length(col) > 0 % i.e. same id radar track exists
          for z=1:length(col)
                rad_pos=tracks(:, col(z));
                rx_tmp= current_gps(1,n)-rad_pos(1);
                ry_tmp= current_gps(2,n)-rad_pos(2);
                xy=[rx_tmp;ry_tmp];
                [rc,cc]=size(rx_corr);
                rx_corr(rad_pos(7)*2-1:rad_pos(7)*2, cc+1)=xy;% append correction to rx_corr matrix
                [rrx,crx]=size(rx_corr);
                if typ == 1
                  for nn=2:2:rrx
                        cs=find(rx_corr(nn,:)~=0);
                        if length(cs)>0
                          correction(1,nn/2)=sum(rx_corr(nn-1,:)) / length(cs);
                          correction(2,nn/2)=sum(rx_corr(nn,:)) / length(cs);
                        end
                  end
                elseif typ == 2
                  cs=find(rx_corr(1,:)~=0);
                  if length(cs)>0
                        correction(1,1)=sum(rx_corr(1,:)) / length(cs);
                        correction(2,1)=sum(rx_corr(2,:)) / length(cs);
                  end
                  if rrx>2
                        tempx=0;
                        tempy=0;
                        num=0;
                        for nn=4:2:rrx
                          cs=find(rx_corr(nn,:)~=0);
                          num=num+length(cs);
                          tempx=sum(rx_corr(nn-1,:)) +tempx;
                          tempy=sum(rx_corr(nn,:)) +tempy;
                  end
```

```
                    corrx=sum(tempx) / num;
                    corry=sum(tempy) / num;
                    corr=[corrx; corry];
                    correction(1:2,2:5)=kron(corr, ones(1,4));
                 end
           end
           disp_gps=[current_gps(:,n);0;0];
           disp('     GPS      Radar ')
           gps_samnam=[disp_gps rad_pos]; disp(gps_samnam)
           disp('Current Correction > (Meters)'), disp(xy.*1852)
           disp('Average Corrections'), disp(correction*1852), pause(2)
        end
  else % do this if no vessel with same id is found
        closest=memb(tracks, current_gps(:,n)); % returns positionally closest radar obs
        rx_tmp= current_gps(1,n)-closest(1);
        ry_tmp= current_gps(2,n)-closest(2);
        disp('Current GPS report is')
        disp(current_gps(:,n))
        disp('Positionally/temporally closest radar report is')
        disp(closest)
        rx_tmp=current_gps(1,n)-closest(1);
        ry_tmp=current_gps(2,n)-closest(2);
        disp('Correction is > (IN METERS)')
        disp('x correction'), disp(rx_tmp*1852)
        disp('y correction'), disp(ry_tmp*1852)
        accept_corr=input('Accept Correction? (1=yes, 0=no) > ');
        if accept_corr==1
           xy=[rx_tmp;ry_tmp];
           [rc,cc]=size(rx_corr);
           rx_corr(closest(7)*2-1:closest(7)*2, cc+1)=xy,
           [r,c]=size(rx_corr);
           for nn=2:2:r
                 cs=find(rx_corr(nn,:)~=0);
                 if length(cs)>0
                    correction(1,nn/2)=sum(rx_corr(nn-1,:)) / length(cs);
                    correction(2,nn/2)=sum(rx_corr(nn,:)) / length(cs);
                 end
           end
        end
        chng= input('Change id of closest radar obs to gps id?, 1 for yes, 0 for no >  ')
        if chng==1
           nams((closest(7)*3-3)+closest(8))=current_gps(7,n);
        end
  end
end
```

```
function [correction, rx_corr, nams] = gpskalm(epoch, track_hist, rx_corr, window, nams)
% function [correction, rx_corr, nams] = gpskalm(epoch, track_hist, rx_corr, window, nams)
% This function updates the correction to be made to each radar.  The radar report closest in time to the
% gps reporting time is used to determine the correction.
% INPUTS: epoch - current epoch in nysim
%        track_hist - history of each track from mass storage
%        rx_corr - (x,y) all previous corrections for each radar
%        window - size of reporting window
%          nams - current identification vector
% Written by LT T. Ruthenberg
typ=2;
load gps1  % contains v_gps generated in nydata1.m
load radarll
radar_radius=[7 3 3 3 2];
current_gps=[];
correction=zeros(2,5);
for n=1:length(v_gps)
  if epoch-window+1 <= v_gps(5,n) & epoch >= v_gps(5,n)
        current_gps=[current_gps v_gps(:,n)];
  end
end

                              % search track history for same identifier track
[r,c]=size(current_gps);
[rh,ch]=size(track_hist);
for n=1:c
 if current_gps(8,n) > 3 % 4 satellites req'd for accurate GPS report
 tracks=[];
  for z=10:10:rh
        row_time=z-5;
        temp=max(find(track_hist(row_time,:) > 0));
        col=max(find(track_hist(row_time,1:temp)<=current_gps(5,n))); % temporal alignment
        if col > 1 & abs(current_gps(5,n)-track_hist(row_time,col)) < 10
          tracks=[tracks track_hist(z-9:z,col)];
        end
  end
                              % search the temporally aligned tracks vector for same name vessel
  col=find(current_gps(7,n)==tracks(9,:));
  if length(col) > 0 % i.e. same id radar track exists
        for z=1:length(col)
          if current_gps(5,n) == tracks(5,col(z))
                rad_pos=tracks(:, col(z));
          else % when epochs are not same
    id_hist=track_hist(tracks(7,col(z))*30-30+tracks(8,col(z))*10-10+1:tracks(7,col(z))*30-30+ ...
          tracks(8,col(z))*10, :);
                id_hist=id_hist(1:9, find(id_hist(1, :) > 0)); [r1,c1]=size(id_hist);
                id=id_hist(9,c1);
                if c1>20,  id_hist=id_hist(:,c1-20:c1); end
                full_id_hist=fillup(id_hist);
                rad_pos=predict(full_id_hist, current_gps(5,n));
                rad_pos=[rad_pos; id; -Inf];
```

```
            end
            rx_tmp= current_gps(1,n)-rad_pos(1);
            ry_tmp= current_gps(2,n)-rad_pos(2);
            xy=[rx_tmp;ry_tmp];
            [rc,cc]=size(rx_corr);
            rx_corr(rad_pos(7)*2-1:rad_pos(7)*2, cc+1)=xy;% append correction to historical rx_corr matrix
            [rrx,crx]=size(rx_corr);

if typ == 1
            for nn=2:2:rrx
                    cs=find(rx_corr(nn,:)~=0);
                    if length(cs)>0
                       correction(1,nn/2)=sum(rx_corr(nn-1,:)) / length(cs);
                       correction(2,nn/2)=sum(rx_corr(nn,:)) / length(cs);
                    end
            end
elseif typ == 2
            cs=find(rx_corr(1,:)~=0);
            if length(cs)>0
                    correction(1,1)=sum(rx_corr(1,:)) / length(cs);
                    correction(2,1)=sum(rx_corr(2,:)) / length(cs);
            end
            if rrx>2
                    tempx=0;
                    tempy=0;
                    num=0;
                    for nn=4:2:rrx
                       cs=find(rx_corr(nn,:)~=0);
                       num=num+length(cs);
                       tempx=sum(rx_corr(nn-1,:)) +tempx;
                       tempy=sum(rx_corr(nn,:)) +tempy;
                    end
                    corrx=sum(tempx) / num;
                    corry=sum(tempy) / num;
                    corr=[corrx; corry];

                    correction(1:2,2:5)=kron(corr, ones(1,4));
            end
end
            disp_gps=[current_gps(:,n);0;0];
            disp('     GPS      Radar ')
            gps_samnam=[disp_gps rad_pos]; disp(gps_samnam)
            disp('Current Correction > (Meters)'), disp(xy.*1852)
            disp('Average Corrections'), disp(correction*1852)
         end
  else                                                          % do this if no vessel with same id is found
         closest=memb(tracks, current_gps(:,n)); % returns positionally closest radar obs
         rx_tmp= current_gps(1,n)-closest(1);
         ry_tmp= current_gps(2,n)-closest(2);
         disp('Current GPS report is')
         disp(current_gps(:,n))
```

```
            disp('Positionally/temporally closest radar report is')
            disp(closest)
            rx_tmp=current_gps(1,n)-closest(1);
            ry_tmp=current_gps(2,n)-closest(2);
            disp('Correction is > (IN METERS)')
            disp('x correction'), disp(rx_tmp*1852)
            disp('y correction'), disp(ry_tmp*1852)
            accept_corr=input('Accept Correction? (1=yes, 0=no) > ');
            if accept_corr==1
              xy=[rx_tmp;ry_tmp];
              [rc,cc]=size(rx_corr);
              rx_corr(closest(7)*2-1:closest(7)*2, cc+1)=xy,
              [r,c]=size(rx_corr);
              for nn=2:2:r
                      cs=find(rx_corr(nn,:)~=0);
                      if length(cs)>0
                        correction(1,nn/2)=sum(rx_corr(nn-1,:)) / length(cs);
                        correction(2,nn/2)=sum(rx_corr(nn,:)) / length(cs);
                      end
              end
            end
            chng= input('Change id of closest radar obs to gps id?, 1 for yes, 0 for no >  ')
            if chng==1
              nams((closest(7)*3-3)+closest(8))=current_gps(7,n);
            end
  end
 end
end
```

```
function track_hist = history(obs, track_hist, nams)
% function track_hist = history(obs, track_hist, nams)
% Function stores all observations based on the track for DGPS processing. (Mass Storage)
% INPUTS: obs - all observations within the reporting window
%         track_hist - history of each radar/track
%         nams - identity associated with each radar/track
% Written by LT T. Ruthenberg, USN
[r,c]=size(obs);
for n=1:c
  row=obs(7,n)*30-29 + obs(8,n)*10-10;
  col=max(find(track_hist(row,:) > -Inf)) + 1;
  track_hist(row:row+7,col)=obs(:,n);
  track_hist(row+8, col)=nams(obs(7,n)*3-3+obs(8,n));
end
```

```
function val = mem(obs, mode)
% function val = mem(obs)
% function is called by odp.m.  This function determines all the membership parameters of two
% observations being compared for fusion during the estimation process.
% INPUTS: obs - observations with the same identity
% Written by LT T. Ruthenberg, USN
[r,c]=size(obs);
mem_track=zeros(2,(c*(c-1))/2);
if mode==0
 for n=1:c-1
  for m=1:2
        if abs(obs(m,n)-obs(m,n+1)) < 500/1852
          x=abs(obs(m,n)-obs(m,n+1));
          mem_track(m,n) = abs(-x*1852/500 + 1);
        else
          mem_track(m,n) = 0;
        end
  end
                              %%%%%%  COURSE %%%%%
   if obs(3,n)>320 & obs(3,n+1)<40   % continuity
        x=360-obs(3,n)+obs(3,n+1);
        mem_track(3,n) = -x/80 + 1;
   elseif obs(3,n)<40 & obs(3,n+1)>320
        x=360-obs(3,n+1)+obs(3,n);
        mem_track(3,n) = -x/80 + 1;
   elseif abs(obs(3,n) - obs(3,n+1)) < 50
     x=abs(obs(3,n)-obs(3,n+1));
        if x > 180, x=360-x; end
        mem_track(3,n) = -x/80 + 1;
   else
     mem_track(3,n) = 0;
   end
                             %%%%%  SPEED  %%%%%%
  if abs(obs(4,n)-obs(4,n+1)) <= 1
        mem_track(4,n) = 1;
  elseif abs(obs(4,n)-obs(4,n+1)) > 1 & abs(obs(4,n)-obs(4,n+1)) < 6
        x=abs(obs(4,n)-obs(4,n+1));
    mem_track(4,n) = -x/5 + 6/5;
  else
        mem_track(4,n) = 0;
  end
 end
else
 for n=1:c-1
  for m=1:2
        if abs(obs(m,n)-obs(m,n+1)) < 500/1852
          x=abs(obs(m,n)-obs(m,n+1));
          mem_track(m,n) = abs(-x*1852/500 + 1);
        else
          mem_track(m,n) = 0;
        end
```

```
   end
  end
end
[r,c]=size(mem_track);
if c == 1
        [val, col]=min(mem_track);
else
        [val, col]=min(min(mem_track));
end
disp('Memberships')
disp(mem_track)
```

function closest = memb(r_hist, current_gps)
% function track_num = memb(r_hist, current_gps)
% This function is called by gpsclos.m or gpskalm.m to determine which radar observations is closest.
% Decision is made based solely on position membership.
% INPUTS: r_hist - most current radar report from each track from mass storage
%   current_gps - a single GPS report
% Written by LT T. Ruthenberg, USN

```
[r,c]=size(r_hist);
mem_track=zeros(2,c);
for n=1:c
  for m=1:2
    if abs(r_hist(m,n)-current_gps(m,1)) < 500/1852
      x=abs(r_hist(m,n)-current_gps(m,1));
      mem_track(m,n) = abs(-x*1852/500 + 1);
    else
      mem_track(m,n) = 0;
    end
  end
end
mem_sum=sum(mem_track);
[val, col]=max(mem_sum);
closest=r_hist(:,col);
```

function [mem_track, thresh_mat] = memship(obs, max_threshold, min_thresh, mode)
%function [mem_track, thresh_mat] = memship(obs, max_threshold, min_thresh, mode)
% This function determines the membership of all observations as compared to each other observation.
% Calculation ends when one parameter is below the variable threshold.
% INPUTS: obs - current observations
%   max_threshold - determines max value of variable threshold function
%   min_threshold - determines min value of variable threshold function
% min & max threshold are not used in this function, but are passed to thresh.m
% Written by LT T. Ruthenberg, USN

```
[r,col] = size(obs);
spd1=1;   % accuracy of full membership in speed
mem_track=NaN .* ones(4*(col),col);
thresh_mat=zeros(col,col);
for p=1:col
ref=obs(:,p);
```

```
for n=1:col
 if p==n
 else
  threshold=thresh(ref, obs(:,n), max_threshold, min_thresh);
  thresh_mat(p,n)=threshold;
                                %%%%%%  POSITION  %%%%%
 for m=1:2
   if abs(obs(m,n)-ref(m,1)) < 500/1852
     x=abs(obs(m,n)-ref(m,1));
     mem_track(4*p-4+m,n) = abs(-x*1852/500 + 1);
   else
     mem_track(4*p-4+m,n) = 0;
   end
   if mem_track(4*p-3,n) < threshold
         break
   end
 end
                                %%%%%%  COURSE %%%%%
 if mode==0 & mem_track(4*p-3:4*p-2,n)>threshold
  if (obs(3,n)>320 & ref(3,1)<40)
         x=360-obs(3,n)+ref(3,1);
         mem_track(4*p-1,n) = -x/80 + 1;
  elseif (obs(3,n)<40 & ref(3,1)>320)
         x=360-ref(3,1)+obs(3,n);
         mem_track(4*p-1,n) = -x/80 + 1;
  elseif abs(obs(3,n) - ref(3,1)) < 50
    x=abs(obs(3,n)-ref(3,1));
         if x > 180, x=360-x; end
         mem_track(4*p-1,n) = -x/80 + 1;
  else
     mem_track(4*p-1,n) = 0;
  end
 else
   mem_track(4*p-1,n) = NaN;
 end
                              %%%%%  SPEED  %%%%%%
 if mode==0 & mem_track(4*p-3:4*p-1,n)>threshold
   if abs(obs(4,n) - ref(4,1)) <= spd1
     mem_track(4*p,n) = 1;
   elseif abs(obs(4,n) - ref(4,1)) > 1 & abs(obs(4,n) - ref(4,1)) < 6
     x=abs(obs(4,n)-ref(4,1));
     mem_track(4*p,n) = -x/5 + 6/5;
   else
     mem_track(4*p,n) = 0;
   end
 else
   mem_track(4*p,n) = NaN;
 end
 end
end
end
```

```
function [obs_n, prosp_list, nams, relate_mat] = names(nams, obs1, prosp_list, relate_mat)
% function [obs_n, prosp_list, nams] = names(nams, obs1, prosp_list, relate_mat)
% Function attaches identifiers to the observations.  First the existing list of identifiers (nams) is checked,
% next the  prospective list and finally the user is prompted if an identity has not been found.
% INPUTS: nams - vector containing current "names" for each radar/track
%       obs1 - current observations
%       prosp_list - current prospective list
%       relate_mat - correlation matrix relating a track with another track
% Written by LT T. Ruthenberg, USN
[r,c]=size(obs1);
obs_n=[obs1; zeros(1,c)];  % attach 9th row for names

for n=1:c
   [rp,cp]=size(prosp_list);

        %%%% check nams vector
   if nams(obs1(7,n)*3-3+obs1(8,n)) ~= 0
        obs_n(9,n)=nams(obs1(7,n)*3-3+obs1(8,n));

        %%%% check correlation matrix (relate_mat)
   elseif sum(relate_mat(obs1(7,n)*3-3+obs1(8,n),:)) > 0
        col=find(relate_mat(obs1(7,n)*3-3+obs1(8,n),:));
        if length(col) == 1             % obs must be correlated with a single radar/track
           nams(obs1(7,n)*3-3+obs1(8,n))=nams(col);  % pass on identifier
           obs_n(9,n)=nams(obs1(7,n)*3-3+obs1(8,n));
        elseif length(col) > 1
           dup=sprintf(' Radar %i track %i is related to more than one vessel', obs1(7,n), obs1(8,n));
           disp(dup);
           obs_n(9,n)=99; % indicates unknown association
           for mm=1:length(col)
                relate_mat(obs1(7,n)*3-3+obs1(8,n), col(mm))=0;
                relate_mat(col(mm), obs1(7,n)*3-3+obs1(8,n))=0;
           end
        end

        %%%% check prospective list

   elseif cp > 0   % if size of prosp_list > 0
        if sum(prosp_list(4,:)) < cp  % if pl contains unused prospectives
           col=plclose(prosp_list, obs1(:,n));
           if col > 0
                nams(obs1(7,n)*3-3+obs1(8,n))=prosp_list(3,col);
                obs_n(9,n)=nams(obs1(7,n)*3-3+obs1(8,n));
                prosp_list(4,col)=1;
                prosp_list1=[];  %%% remove used members of prospective list
                for n=1:cp
                   if prosp_list(4,n)==0
                        prosp_list1=[prosp_list1 prosp_list(:,n)];
                   end
                end
                prosp_list=prosp_list1;
```

```
            else
                  obs_n(9,n)=prompt(obs1(:,n));
                  nams(obs1(7,n)*3-3+obs1(8,n))=obs_n(9,n);
            end
         end

         %%%% prompt operator for name of vessel
   else
         disp('Prompting due to no name or PL from names.m')
         obs_n(9,n)=prompt(obs1(:,n));
         nams(obs1(7,n)*3-3+obs1(8,n))=obs_n(9,n);
   end
end

for n=1:c
   if obs_n(9,n)==0
         obs_n(9,n)=prompt(obs1(:,n));
         nams(obs_n(7,n)*3-3+obs_n(8,n))=obs_n(9,n);
   end
end
```

% nysim.m
% This file is the main program for the nysim simulation. This file calls all of the algorithm functions.
% Written by LT T. Ruthenberg, USN

```
load rad_loc
load obs41  % created in raddat1.m, loads rad_obs
load gps1  % created in nydata1.m, loads v_gps
                    %%%%% PERFORMANCE PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
window=5;          % window size
assn_wt=4;         % association weight
assn_max=8;        % max association weight
max_thresh=0.9;% maximum threshold
min_thresh=0.5;% minimum threshold
abs_min_thresh=0.4 ; %absolute minimum threshold (for fusion)
loss_rate=0.3;     % loss rate of radar observations
rpt_wt=2;
mode=0;  % mode=0:use pos, course & speed;  mode=1 use pos only
                    % % % % % % % % % % % % % %         I N I T I A L I Z A T I O N
%%%%%%%%%%%%%%%%%%%%%%%%%
gps_correction=zeros(2,5); rp=0; cp=0;
relate_mat=zeros(15); % size determined by num. of radar & tracks per radar
rpt_rad=zeros(3,5);  % size is tracks per radar X # of radars
nams=zeros(1,15);
prosp_list=[13 8 4 3 6 6 10.03 10; 4 6 11 14 12 16 5.03 5; 1 2 5 4 3 6 7 8; 0 0 0 0 0 0 0 0];
odp_plot=[];
max_its=550;
track_hist=[zeros(150,1) -Inf.*ones(150, max_its)];
rx_corr=[];
a=[];
```

```
%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=window:window:max_its
  obs=currobs(rad_obs(:,n-(window-1):n), loss_rate);          % currobs req only for sim, filters out NaN obs
  obs_corr=correct(obs, gps_correction); %gps correct all obs for history % DGPS correction, preprocessing
  track_hist=history(obs_corr, track_hist, nams); % saves track history  %save to mass storage
  obs1=rem_old(obs_corr); % returns most current obs in window        % alignment
  [ro,co]=size(obs1);
  for no=1:co  % set MZ flag according to mode        % set MZ flag according to desired mode
        obs1(6,no)=mode;
  end
   [rpt_rad, nams, prosp_list]=rpt(rpt_rad,obs1, nams, rpt_wt, track_hist, prosp_list); % clear non-reporting
                                                                                      % identifiers
  [mems, thresh_mat]=memship(obs1, max_thresh, min_thresh, mode);  % determine memberships of obs pairs
  assc=assoc(mems, thresh_mat);                                   % generate association matrix
  [relate_mat, nams]=relate(obs1, assc, relate_mat, nams, assn_wt, assn_max, rpt_rad); %track correlation matrix
  [obsn, prosp_list, nams, relate_mat]=names(nams, obs1, prosp_list, relate_mat); % attach identifiers
  [odp_obs, nams]=odp(obsn, nams, abs_min_thresh, thresh_mat, relate_mat, mode);% obs with same identity
                                                                                  %are fused
 check_gps=ch_epoc(n, window);  % necessary for simulation only, to determine if DGPS report in this window
  if check_gps==1
        [gps_correction, rx_corr, nams]=gpsclos(n, track_hist, rx_corr, window, nams);
  end
end
```

function [odp_obs, nams] = odp(obs, nams, abs_min_thresh, thresh_mat, relate_mat, mode)
% function [odp_obs, nams] = odp(obs, nams, abs_min_thresh)
% This function fuses redundant observations. Fusion is based on the identity of the observations. The
% centroid of the observations with the same identity is calculated for display. If the track correlation
% matrix for the same id pair is greater than zero then the absolute minimum is used for fusion, otherwise
% the variable minimum is used.
% INPUTS: obs - current observations
%   nams - vector containing current identities for each track
%   abs_min_thresh - absolute minimum threshold, to fuse obs with same name
% Written by LT T. Ruthenberg, USN

```
[r, c]=size(obs);
used=zeros(1,c);
odp_obs=[];
fuse_obs=[];
for n=1:c
  if used(n)==0
        temp=[];
        for m=1:c
           if obs(9,n)==obs(9,m)
                 temp=[temp obs(:,m)]; % group same id obs
                 used(n)=1; used(m)=1;
                 var_thresh=thresh_mat(n,m);
           end
     end
        [rt,ct]=size(temp);
        if ct==1
           fuse_obs=temp; % if single obs
```

```
elseif ct < 3
  row=temp(7,1)*3-3+temp(8,1);
  col=temp(7,2)*3-3+temp(8,2);
  if relate_mat(row,col) > 0
        min_thresh=abs_min_thresh;
        disp('historical fusion allowed')
        disp(' ')
  else
        min_thresh=var_thresh;
        disp('Historical correlation = 0, therefore reduced threshold not allowed')
        disp(' ')
        st=sprintf(' Variable threshold = %5.2f ', min_thresh);
        disp(st)
  end
  min_mem=mem(temp, mode);
  sm=sprintf('Minimum membership = %5.2f ', min_mem);
  if min_mem > min_thresh
        fuse_obs=(sum(temp(1:4,:)') ./ ct)';
        if fuse_obs(3)>360
          fuse_obs(3)=fuse_obs(3)-360
        end
        fuse_obs=[fuse_obs; temp(5:9,ct)];
        disp('     Fusing'), disp(temp)
        disp('--------------------------------------')
  else
        dist=sqrt((temp(1,1)-temp(1,2))^2 + (temp(2,1)-temp(2,2))^2)*1852;
        disp('ERROR ATTEMPTING TO FUSE OBS GREATER THAN MAX SEPARATION
            ALLOWED')
        s1=sprintf(' Actual Distance = %5.2f meters', dist);
        disp(s1), disp(temp)
        f=input('Fuse anyway? (1 for yes, 0 for no) >>  ');
        if f==1
          fuse_obs=(sum(temp(1:4,:)') ./ ct)';
          if fuse_obs(3)>360
                fuse_obs(3)=fuse_obs(3)-360
          end
          fuse_obs=[fuse_obs; temp(5:9,ct)];
        else
          for z=1:ct
            temp(9,z)=prompt(temp(:,z));
            nams(temp(7,z)*3-3+temp(8,z))=temp(9,z);
            fuse_obs=[fuse_obs temp(:,z)];
          end
        end
  end
else
  for z=1:ct
        temp(9,z)=prompt(temp(:,z));
        nams(temp(7,z)*3-3+temp(8,z))=temp(9,z);
        fuse_obs=[fuse_obs temp(:,z)];
  end
```

```
            end
            odp_obs=[odp_obs fuse_obs];
    end
end
```

```
function col = plclose(prosp_list, obs)
% function col = plclose(prosp_list, obs)
% This function is called by names.m. This function determines which unused prospective vessel is closest
% to the observation, and within the gate (1.5 nm).
% INPUTS: prosp_list - current prospective list
%         obs - a single observation
% OUTPUT: col - column of appropriate prosp_list component
% Written by LT T. Ruthenberg, USN
pl=[];
[r,c]=size(prosp_list);
col=0;
for n=1:c
   if prosp_list(4,n)==0  % remove used prosp_list members
            pl=[pl prosp_list(:,n)];
   end
end
[r,c1]=size(pl);
dist=zeros(1,c1);
for n=1:c1
   dist(n)=sqrt((pl(1,n)-obs(1))^2 + (pl(2,n)-obs(2))^2);
end
[val,col1]=min(dist); % determines column of prosp_list with min dist from obs
if val < 1.5
   for n=1:c  %% find the matching column in prosp_list
            if pl(3,col1)==prosp_list(3,n)
               col=n;
               break;
            end
   end
end
```

```
function xhat1 = predict(dbt, gps_time)
% function xhat1 = predict(dbt, gps_time)
% This function is a Kalman filter. The function is called by gpskalm.m to estimate radar position at the
% gps time
% INPUTS: dbt - last 30 seconds of radar/track history
%         gps_time - time to which filter must estimate radar position
% OUTPUT: xhat1 - estimated position at the gps_time
% Written by LT T. Ruthenberg, USN
dbt_max=max(dbt(5,:));
num_pred=gps_time-dbt_max;
T=3;      %T is in seconds
[row,col]=size(dbt);
kdbt=xdotydt(dbt); %sends [x;y;crs;spd] existing history of 1 track
phi=[1 T/3600 0 0;0 1 0 0; 0 0 1 T/3600;0 0 0 1]; % assuming no acceleration
                                                % obtained from X(k+1) = phi*x(k)
```

```
R=[(100/1852)^2/12 0 0 0;0 1/12 0 0; 0 0 (100/1852)^2/12 0; 0 0 0 1/12]; % R is measurement error variance,
                                                                          % values are assumed
Q=0.5.* R; % Q helps maneuver tracking     % parameters for Q & R dependent on update rate
%%%%%%%%%%%%%%%%%%%% initial conditions %%%%%%%%%%%%%%%%%%%%%%%%%%%
Xhat=kdbt(:,1);  % set to first (oldest) obsevation
P=[8/1852 0 0 0; 0 .2 0 0; 0 0 8/1852 0; 0 0 0 .1]; % var[x(t0)]
K=[0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
%%%%%%%%%%%%%%%%%%%%%%%%% recursive solution for Kalman gains %%%%%%%%%%%
for n=1:col+num_pred
   %% predicted state ( prediction based on estimates)
         Xhat(:,n+1)=phi*Xhat(:,n);
         if n==col+num_pred, break; end
   %%%%***%%%% this portion added for prediction  %%%%***%%%%
         if n>=col
            kdbt=[kdbt Xhat(:,n+1)];
         end
   %% Predicted estimate for Z ( prediction based on noisy obs)
         Zhat=Xhat(:,n+1);
   %% Linear filter of state vector recall z=H*x + v
         H=eye(4); % assuming filter to be a unity filter
   %% Predicted covariance matrix
         P=phi*P*phi'+Q;
   %% Kalman gain eqn
         K=P*H'*inv(H*P*H'+R);
   %% corrected covariance matrix
         P=(eye(4)-K) * P;
   %% correction eqn
         Xhat(:,n+1)=Xhat(:,n+1) + K*(kdbt(:,n+1)-Zhat);
end;
xhat=xdot2xy(Xhat); %converting back to [x;y;crs;sp;...]
time=dbt(5,1):dbt(5,1)+length(xhat)-1;
xhat=[xhat; time];
mz_rad_tr=kron([dbt(6,col) dbt(7,col) dbt(8,col)]', ones(1,length(xhat)));
xhat=[xhat; mz_rad_tr];
[r,c]=size(xhat);
xhat1=xhat(:,c); % predicted position
```

```
function nam = prompt(obs)
% function nam = prompt(obs)
% This function prompts the operator for identity of an unknown vessel.
% INPUT: obs - a single observation
% OUTPUT: nam: identity to be assigned to the observation
% Written by LT T. Ruthenberg, USN
obs
s1=sprintf('Enter name of vessel entering near x = %5.2f\n', obs(1));
s2=sprintf('                          y = %5.2f\n', obs(2));
s3=sprintf('                        crs = %5.2f\n', obs(3));
s4=sprintf('                        spd = %5.2f\n', obs(4));
s5=sprintf('                      radar = %d', obs(7));
disp(s1), disp(s2), disp(s3), disp(s4), disp(s5)
nam=input('                           >> ');
```

```
function new_obs = rem_old(obs)
% function new_obs = rem_old(obs)
% function removes old versions of same track that are within the reporting window, and returns the most
% current observation % for each radar/track.
% INPUTS: obs - all observations within the reporting window
% Written by LT T. Ruthenberg, USN
[r,c]=size(obs);
new_obs=[];
used=ones(1,c);
for n=1:c-1
   ref=obs(:,n);
   if used(n)==1
         new_obs=[new_obs ref];
   end
   used(n)=0;
   [row, col]=size(new_obs);
         for m=n+1:c
            if ref(7:8)==obs(7:8,m) & obs(5,m)>ref(5) & used(m)==1
                  new_obs(:,col)=obs(:,m);  % if the obs is from the
                  used(m)=0;                % same radar and same track
                                            % and is more recent, then
            end                       % replace the ref in new_obs
         end
end
a=find(used);
for n=1:length(a)
   new_obs=[new_obs obs(:, a(n))];
end

function [new_rpt_rad, nams, prosp_list] = rpt(rpt_rad, obs1, nams, rpt_wt, track_hist, prosp_list)
% function [new_rpt_rad, nams] = rpt(rpt_rad, obs1, nams)
% Purpose of this function is to remove an identity from the nams vector of a track which has not reported
% in the % selected reporting weight number of windows
% INPUTS: rpt_rad - matrix of reporting radar/tracks
%         obs1 - current observations
%         nams - vector containing current identifiers for each radar/track
%        rpt_wt -  report weight- number of non-reporting windows before identifier is cleared
% Written by LT T. Ruthenberg
new_rpt_rad=zeros(3,5);
[r,c]=size(obs1);
for n=1:c                         % set current rpt_rad to rpt_wt
   new_rpt_rad(obs1(8,n), obs1(7,n))=rpt_wt;
end
[r,c]=find(rpt_rad);
for n=1:length(r)                 % subtract 1 from old rpt_rad values > 0
   rpt_rad(r(n),c(n))=rpt_rad(r(n),c(n))-1;
end
new_rpt_rad=new_rpt_rad+rpt_rad;
[r1,c1]=find(new_rpt_rad > rpt_wt);          % limit max value of rpt_rad to rpt_wt
for n=1:length(r1)
   new_rpt_rad(r1(n), c1(n))=rpt_wt;
```

```
end
col=find(nams);
for n=1:length(col)
   radar=ceil(col(n)/3);
   track=col(n) - (radar*3-3);
   if new_rpt_rad(track, radar)==0              % clear identity of tracks whose value is zero
         nams(col(n))=0;
         disp('Clearing Nams vector due to no radar update')
         disp(col(n))
         if radar==1
            hst=track_hist(radar*10-10+track*10-9:track*10, :);
            hst_col=find(hst(9,:)>0);
            hst=hst(:, min(hst_col):max(hst_col));
            [rh, ch]=size(hst);
            obs_last=hst(:, ch);
            if obs_last(3)>270 | obs_last(3)<90 & obs_last(2)>9
                  pl=[5, 11.5, obs_last(9), 0]'; % automatic PL list entry when
                  prosp_list=[prosp_list pl];    % departing radar 1 to the north
            end                                           % or radar 3 to the south
         end
   end
end
```

function mod_threshold = thresh(refl, obs, max_t, min_t)
% function mod_threshold = thresh(refl, obs, max_t, min_t)
% This function creates the variable threshold based on position of either the reference observation or the
% observation being compared to. This function is called by memship.m. A piecewise linear function is
% generated from max_t & min_t
% INPUTS: refl - a single observation
% obs - a single observation
% max_t - maximum threshold
% min_t - minimum threshold
% Written by LT T. Ruthenberg, USN

```
load rad_loc % contains radar locations in meters (for simulation)
dist_ref=sqrt((rad_loc(1,refl(7))-refl(1))^2 + (rad_loc(2,refl(7))-refl(2))^2);
dist_obs=sqrt((rad_loc(1,obs(7))-obs(1))^2 + (rad_loc(2,obs(7))-obs(2))^2);
if dist_ref<=1 & dist_obs<=1
   mod_threshold=min_t;
elseif (dist_ref > 1 | dist_obs > 1) & (dist_ref < 5 & dist_obs < 5)
   mod_threshold=((max_t-min_t)/4)*min([dist_ref dist_obs]) + min_t-((max_t-min_t)/4);
else
   mod_threshold=max_t;
end
if mod_threshold < min_t
   mod_threshold=min_t;
end
```

APPENDIX C: SIMULATION CODE

```
function y = ch_epoc(epoch, window)
% function y = ch_epoc(epoch)
% This function determines whether a DGPS report occurs within the reporting window
% Written by LT T. Ruthenberg, USN
gps_epochs=[48 70 120 168 190 240 288 310 360 408 430 480 528];
y=0;
for n=1:length(gps_epochs)
   if epoch-window < gps_epochs(n) & epoch>=gps_epochs(n)
         y=1;
   end
end

function crs = course(x,y)
% function crs = course(x,y)
% This function is called by fcreate.m.  The function determines the
% course in degrees for a given observation.
% INPUTS: x - delta x
%         y - delta y
% Written by LT T. Ruthenberg, USN
if x >= 0                          % handles quad 1&4, 000, 090, 180
   crs = 90 - atan(y/x)*180/pi;
elseif x < 0                       % handles quad 2&3, 270
   crs = 270 - atan(y/x)*180/pi;
end

function tracks = createny(lon1, lat1, lon2, lat2, spd, st_time, inc)
% function creates clean position reports based on given start and stop positions and speed
%Tthe generated track will consist of [x, y, crs, spd, time]'
% INPUTS: lon1/lat1 - long/lat to start from
%         lon2/lat2 - long/lat to create track to
%         spd - desired speed
%         st_time - start time, needed for time (epoch) stamp
%         inc - increments, radar report rate (3 seconds for this simulation)
% Written by LT T. Ruthenberg, USN
load ref
[x, y]=ll2nm(lon1, lat1, lon2, lat2);
dist=sqrt(x^2+y^2);
crs=course(x,y);
[x,y]=ll2nm(ref(1), ref(2), lon1, lat1);  % starting place
time=st_time;
for n=1:floor(dist*3600/(inc*spd))
   if crs >= 0 & crs <=90
         x1=x+cos(radians(90-crs))*spd*inc/3600;
         y1=y+sin(radians(90-crs))*spd*inc/3600;
   elseif crs > 90 & crs <=180
         x1=x+cos(radians(crs-90))*spd*inc/3600;
         y1=y-sin(radians(crs-90))*spd*inc/3600;
   elseif crs > 180 & crs <= 270
         x1=x-cos(radians(270-crs))*spd*inc/3600;
```

```
        y1=y-sin(radians(270-crs))*spd*inc/3600;
  elseif crs > 270
        x1=x-cos(radians(crs-270))*spd*inc/3600;
        y1=y+sin(radians(crs-270))*spd*inc/3600;
  end
  tracks(1,n)=x1;
  tracks(2,n)=y1;
  tracks(3,n)=crs;
  tracks(4,n)=spd;
  time=time+1;
  tracks(5,n)=time;
  x=x1;
  y=y1;
end
```

function [lon, lat] = km2lonlat(lon_orig,lat_orig,east,north)
%KM2LONLAT Convert distances in km referenced to a lon/lat point to lon/lat.
% This function will convert distances in kilometers east and west of a reference longitude/latitude point to longitude/latitude. The equation used is from Bowditch's book "The American Practical Navigator."
% Usage:
% [LON,LAT]=KM2LONLAT(LON_ORIG,LAT_ORIG,EAST,NORTH)
% Inputs:
% LON_ORIG - reference longitude.
% LAT_ORIG - reference latitude.
% EAST - distance east (km) of reference point (scalar or vector).
% NORTH - distance north (km) of reference point (scalar of vector).
% Outputs:
% LON - longitude
% LAT - latitude
% Mike Cook - NPS Oceanography Dept. - FEB 94
% Mike Cook - JUN 94 - added more documentation and error checking.

```
% Check for the correct number of inputs.
        if nargin ~= 4
          error(' You *MUST* supply 4 input arguments ')
        end
        con = radians(lat_orig);
        ymetr = 111132.09 - 566.05 .* cos( 2 .* con) ...
         + 1.2 .* cos(4 .* con) - 0.002 .* cos(6 .* con);
        xmetr = 111415.13 .* cos(con) - 94.55 .* cos(3 .* con) ...
         + 0.012 .* cos(5 .* con);
        lon = east .* 1000 ./ xmetr + lon_orig;
        lat = north .* 1000 ./ ymetr + lat_orig;
```

```
function [east, north] = lonlat2km(lon_orig,lat_orig,lon,lat)
%LONLAT2KM  Convert lat/lon to distances (km) referenced to a lon/lat point.
%  This function will convert longitude/latitude pairs to distances in kilometers east and west of a
% reference longitude/latitude point. The equation used is from Bowditch's book "The American Practical
% Navigator."
% Usage:
%  [EAST,NORTH]=LONLAT2KM(LON_ORIG,LAT_ORIG,LON,LAT)
% Inputs:  lon_orig - reference longitude.
%          lat_orig - reference latitude.
%          lon      - longitude scalar or vector.
%          lat      - latitude scalar or vector.
% Outputs: east     - distance east from reference point (km)
%          north    - distance north from reference point (km)
%          Mike Cook - NPS Oceanography Dept. - FEB 94
%       Mike Cook - JUN 94 - added more documentation and error checking.
          if nargin ~= 4
            error(' You *MUST* supply 4 input arguments ')
          end
          con = radians(lat_orig);
          ymetr = 111132.09 - 566.05 .* cos(2 .* con) + 1.2 ...
           .* cos(4 .* con) - 0.002 .* cos(6 .* con);
          xmetr = 111415.13 .* cos(con) - 94.55 .* cos(3 .* con) ...
           + 0.012 .* cos(5 .* con);
      east = (lon - lon_orig) .* xmetr ./ 1000;
      north = (lat - lat_orig) .* ymetr ./ 1000;

function [x, y] = ll2nm(ref_lon, ref_lat, lon, lat)
% function [x, y] = ll2nm(ref_lon, ref_lat, lon, lat)
% Function determines distance in nautical miles between two lat/long fixes
% INPUTS: ref_lon / ref_lat - first lat/long fix
%       lon / lat - second lat/long fix
% Written by LT T. Ruthenberg, USN
[x, y]=lonlat2k(ref_lon, ref_lat, lon, lat);
x=-x/1.852;
y=y/1.852;

function [lon, lat] = nm2ll(x,y)
% function [lon, lat] = nm2ll(x,y)
% Function converts nautical miles to a lat/long fix for nysim.m.
% INPUTS: x,y - x & y position in nysim simulation
% Written by LT T. Ruthenberg, USN
ref=[74+10/60 40+25/60];
x=x*1.852;
y=y*1.852;
[lon, lat]=km2lonla(ref(1), ref(2), -x, y);
```

```
function noisy_obs = noise(obs)
% function noisy_obs = noise(obs)
% This function is called by raddat1.m to add different levels of noise to an observation depending on the
% observations distance from the reporting radar
% INPUT: obs - a single observation
% Written by LT T. Ruthenberg, USN
load rad_loc
dist=sqrt((rad_loc(1,obs(7))-obs(1))^2 + (rad_loc(2,obs(7))-obs(2))^2);
if dist <= 1
  noise_pos=300;  % measurement noise in meters
elseif dist > 1 & dist < 5
  noise_pos=(-200/4)*dist + 350;
else
  noise_pos=100;
end

done=0;
while done == 0
  x_rand=rand-0.5;
  y_rand=rand-0.5;
  noise_x=noise_pos*x_rand;
  noise_y=noise_pos*y_rand;
  if sqrt(noise_x^2 + noise_y^2) <= noise_pos/2
        done=1;
  end
end
noise_x=noise_x/1852;  % x noise variance in NM
noise_y=noise_y/1852;  % y noise variance in NM
noise_crs=30*(rand-0.5);  % course error in degrees
noise_spd=3*(rand-0.5);   % speed error in knots
noise_all=[noise_x; noise_y; noise_crs; noise_spd];
noisy_obs=obs(1:4)+noise_all;
noisy_obs=[noisy_obs; obs(5:8)];

% nydata1.m
% This file creates the radar tracks for the nysim.m simulation
% Run raddat1.m after this file to create the observations for each radar
% Written by LT T. Ruthenberg, USN
!rm dbt1.mat
!rm gps1.mat
zim                  % loads zimll & zimsp
[row,col]=size(zimll);
                              %%%%% ZIM LIVORNO (track of actual vessel)
st_time=0; inc=3;
temp=createny(zimll(1,1), zimll(1,2), zimll(2,1), zimll(2,2), zimsp(1), st_time, inc);
z=temp;
for n=3:row
  [r, c]=size(temp);
  [lon, lat]=nm2ll(temp(1,c), temp(2,c));
  temp=createny(lon, lat, zimll(n,1), zimll(n,2), zimsp(n-1), temp(5,c), inc);
  z=[z temp];
```

```
end
z=kalmny(z); z=[z; 1:length(z)];
%%%%%%%%%%%% Create crossing track %%%%%%%%%%%%%%%%
[lon1, lat1]=nm2ll(4.3, 11);
[lon2, lat2]=nm2ll(5.1474, 13.9548);
t1a=createny(lon1, lat1, lon2, lat2, 13, st_time, inc);
[r,c]=size(t1a);
[lon1, lat1]=nm2ll(t1a(1,c), t1a(2,c));
[lon2, lat2]=km2lonla(74+1/60+5.53/3600, 40+41/60+18.6/3600, 2.4, 2);
t1b=createny(lon1, lat1, lon2, lat2, 10.0, t1a(5,c), inc);
t1=[t1a t1b];
[r, c]=size(t1);
[lon1, lat1]=nm2ll(t1(1,c), t1(2,c));
[lon2, lat2]=km2lonla(73+58/60+22/3600, 40+42/60+33.9/3600, .68, .68);
t2=createny(lon1, lat1, lon2, lat2, 10.6, t1(5,c), inc);

[r, c]=size(t2);
[lon1, lat1]=nm2ll(t2(1,c), t2(2,c));
[lon2, lat2]=km2lonla(73+58/60+22/3600, 40+42/60+33.9/3600, -.34, 1.854);
t3=createny(lon1, lat1, lon2, lat2, 10.3, t2(5,c), inc);
t=[t1 t2 t3];
t=kalmny(t); t=[t; 1:length(t)];
                              %%%%%%  create "parallel" dgps track  %%%%%
for n=120:120:length(z)
   if n < 886; % 886 is z components in radar 1 coverage area
         z_gps(:,n/120)=z(:,n);
         z_gps(1,n/120)=z_gps(1,n/120)-(250/1852);  % creates 250 meter error to west
   else
         z_gps(:,n/120)=z(:,n);
         z_gps(2,n/120)=z_gps(2,n/120)+(250/1852);  % creates 250 meter error to north
   end
end
                              %%%%% create head-on track %%%%%
v5=createny(zimll(5,1), zimll(5,2), zimll(2,1), zimll(2,2), 14, st_time, inc);
                              %%%%% create slow track %%%%%%%
v6=createny(zimll(5,1), zimll(5,2), zimll(6,1), zimll(6,2), 3, st_time, inc);
v6=v6(:,1:290);
v6(5,:)=(238+5:237+290+5);
gap=zeros(5,22);
v78=[v5(:,1:220) gap v6]; % need to be on same track due to max of 3 vessels per radar
                              %%%%  break z into multiple vessels (A,B,C and D)
v1=z(:, 1:530);
v1_gps=z_gps(:,1:4);
v2=z(:, 531:1032);
v2(5,:)=1:length(v2);  % change epoch times
v2_gps=z_gps(:,5:8);
v2_gps(5,:)=z_gps(5,5:8)-length(v1); % change epoch times
v3=z(:, 1033:1512);
v3(5,:)=1:length(v3);  % change epoch times
v3_gps=z_gps(:,9:12);
v3_gps(5,:)=z_gps(5,9:12)-length(v1)-length(v2);  % change epoch times
```

```
v4=z(:, 1513:length(z));
v4(5,:)=1:length(v4);  % change epoch times
v4_gps=z_gps(:,13:17);
v4_gps(5,:)=z_gps(5,13:length(z_gps))-length(v1)-length(v2)-length(v3);
v_gps=[v1_gps v2_gps v3_gps v4_gps];
[r,c]=size(v_gps);
v_gps=[v_gps; zeros(1,c)];% add MZ row
gps_name=[1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 4]; % vessel names A - D
sats=[7 6 3 4 5 6 7 7 7 6 7 6 7 6 5 6 7];
v_gps=[v_gps; gps_name; sats];
                                %%%% add noise to gps track
noise_pos=7.07/1852;  % position error in meters IAW 1992 Radionavigation Plan
noise_crs=10;  % course error in degrees
noise_spd=1;   % speed error in knots
noise_gps=[noise_pos; noise_pos; noise_crs; noise_spd];

for n=1:c
   v_gps(1:4, n)=v_gps(1:4,n)+(rand(4,1)-0.5*ones(4,1)).*noise_gps;
end
t1=t(:, 1:555); % break t into vessel E & F
t2=t(:, 585:1139);
t2(5,:)=  1:length(t2);
save dbt1 v1 v2 v3 v4 v78 t1 t2
save gps1 v_gps
```

% plot1.m
% this program plots the vessels, simulating an ODP display

```
figure(1), clf
v=[-5 15 -5 20];
axis(v);
 axis('square')
hold on
plot(rad_loc(1,:), rad_loc(2,:), 'w*')
hold on
plot(odp_obs(1,:), odp_obs(2,:), 'w.'), pause(0.1)
[rn, cn]=size(odp_obs);
for nn=1:cn
   if odp_obs(9,nn)==1, a='A';
   elseif odp_obs(9,nn)==2, a='B';
   elseif odp_obs(9,nn)==3, a='C';
   elseif odp_obs(9,nn)==4, a='D';
   elseif odp_obs(9,nn)==5, a='E';
   elseif odp_obs(9,nn)==6, a='F';
   elseif odp_obs(9,nn)==7, a='G';
   elseif odp_obs(9,nn)==8, a='H';
   end
   text(odp_obs(1,nn), odp_obs(2,nn), a); pause(0.1)
   rca;
end
hold off
```

% raddat1.m
% Run nydat1.m prior to this file to create the track observations and gps. This file create observations
% for each radar from the vessel paths created in nydata1.m. This file checks which radars can "see" the
% posiion of each observation, and creates an observation for each radar that can "see" that position
% Written by LT T. Ruthenberg USN

```
%!rm obs41.mat
load dbt1  % created by nydata1.m
% v1 v2 v78 v4 v4 t1 v3 t2 t1 v3 t2
v_lengths=[530 502 532 625 625 555 480 555 555 480 555];
boat=zeros(35,625);
boat(1:5,1:length(v1))=v1;
boat(6:10,1:length(v2))=v2;
boat(11:15,1:length(v3))=v3;
boat(16:20,1:length(v4))=v4;
boat(21:25,1:length(v78))=v78;
boat(26:30,1:length(t1))=t1;
boat(31:35,1:length(t2))=t2;
radarx=[74+44.89/3600 40+28/60+15.37/3600;                 % Sandy Hook
        74+9/60+43.7/3600 40+38/60+27.08/3600;            % Mariners Harbor
        74+5/60+25.33/3600 40+38/60+48.52/3600;           % Bank Street
        74+1/60+5.53/3600 40+41/60+18.6/3600;             % Governors Island
        73+58/60+22/3600 40+42/60+33.9/3600];             % Brooklyn Navy Yard
boat_num=[1 2 5 4 4 6 3 7 6 3 7];
rad_radius=[7 7 7 3 3 3 3 3 3 3 2];
rad=[1 1 1 2 3 3 3 4 4 4 5];
track=[1 2 3 1 1 2 3 1 2 3 1];
rad_obs=zeros(88,625);

for n=1:11
  for m=1:v_lengths(n)
        [lon_obs, lat_obs]=nm2ll(boat(boat_num(n)*5-4,m), boat(boat_num(n)*5-3,m));
        [x, y]=ll2nm(radarx(rad(n),1), radarx(rad(n),2), lon_obs, lat_obs);
        if sqrt(x^2+y^2) < rad_radius(n)
           rad_obs(8*n-7:8*n-3,m)=boat(boat_num(n)*5-4:boat_num(n)*5,m);
        end
  end
  rad_obs(8*n-1:8*n,1:v_lengths(n))=kron([rad(n);track(n)], ones(1,v_lengths(n)));
end
[r,c]=size(rad_obs);
for n=8:8:r
  rad_obs(n-7:n,:)=mz(rad_obs(n-7:n,:)); % set mz flag
end
save clean rad_obs
                        %%%% add noise to rad_obs
for n=8:8:r
  for m=1:c
        if rad_obs(n-7:n-4,m) ~= zeros(4,1)
           nf=rad_obs(n-7:n, m);
           rad_obs(n-7:n, m)=noise(rad_obs(n-7:n, m));
           dist=(sqrt((nf(1)-rad_obs(n-7, m))^2 + (nf(2)-rad_obs(n-6, m))^2))*1852;
```

```
            if dist > 150, disp('Distance greater than 150'), pause, end
          end
  end
end
load z
v2add=z(:,1074:1156);  % add vessels B Bank Street portion
v2add(5,:)=543:length(v2add)+542;
v2add=[v2add; zeros(3,length(v2add))];
v2add(7:8,:)=kron([3;3], ones(1,length(v2add)));
                                                    % add noise
for n=1:length(v2add)
  v2add(:,n)=noise(v2add(:,n));
end

v2add=mz(v2add);  % set mz flag
rad_obs(49:56, 543:625)=v2add;
for n=3:8:83  % after noise is added, check for course outside (0-360)
  for c=1:625
        if isnan(rad_obs(n,c))
        else
          if rad_obs(n,c)<0
                rad_obs(n,c)
                rad_obs(n,c)=rad_obs(n,c)+360;
                rad_obs(n,c)
          elseif rad_obs(n,c)>360
                rad_obs(n,c)
                rad_obs(n,c)=rad_obs(n,c)-360;
                rad_obs(n,c)

          end
        end
  end
end
save obs4s rad_obs
```

function y = srny(x)
% This function creates the simulated New York harbor.
% Written by LT T. Ruthenberg USN

```
rad_radius=[7 3 3 3 2];
ref = [40+25/60 74+10/60];
brooklyn_yard = [40+42/60+33.9/3600 73+58/60+22/3600];
gov_island = [40+41/60+18.6/3600 74+1/60+5.53/3600];
mariners_harbor = [40+38/60+27.08/3600 74+9/60+43.7/3600];
bank_street = [40+38/60+48.52/3600 74+5/60+25.33/3600];
sandy_hook = [40+28/60+15.37/3600 74+44.89/3600];
load rad_loc
v=[-5 15 -5 20];
figure(1)
axis(v);
axis('square'), hold on
```

```
plot(rad_loc(1,:), rad_loc(2,:), 'w*')

                                  %%%% SANDY HOOK
sh1x=linspace(rad_loc(1,1)-rad_radius(1), rad_loc(1,1)+rad_radius(1));
sh1y=abs(sqrt(rad_radius(1)^2-(sh1x-rad_loc(1,1)).^2) + rad_loc(2,1));
plot(sh1x, sh1y, 'y')
plot(sh1x, -sh1y+2*rad_loc(2,1), 'y')

                                  %%%% BANK STREET
bsx=linspace(rad_loc(1,3)-rad_radius(3), rad_loc(1,3)+rad_radius(3));
bsy=abs(sqrt(rad_radius(3)^2-(bsx-rad_loc(1,3)).^2) + rad_loc(2,3));
bsy2=-bsy(60:100)+2*rad_loc(2,3);
bsx3=linspace(rad_loc(1,3), bsx(60), 2);
bsy3=linspace(rad_loc(2,3), bsy2(1), 2);
bsy4=-bsy(1:5)+2*rad_loc(2,3);
bsx5=linspace(bsx(5), rad_loc(1,3), 2);
bsy5=linspace(-bsy(5)+2*rad_loc(2,3), rad_loc(2,3), 2);
plot(bsx, bsy, 'r')
plot(bsx(60:100), bsy2, 'r')
plot(bsx3, bsy3, 'r')
plot(bsx(1:5), bsy4, 'r')
plot(bsx5, bsy5, 'r')

                                  %%%% GOVERNORS ISLAND
gix=linspace(rad_loc(1,4)-rad_radius(4), rad_loc(1,4)+rad_radius(4));
giy=abs(sqrt(rad_radius(4)^2-(gix-rad_loc(1,4)).^2) + rad_loc(2,4));
gi1x=[rad_loc(1,4) rad_loc(1,4)];
gi1y=linspace(-giy(50)+2*rad_loc(2,4), rad_loc(2,4), 2);
gi2x=[rad_loc(1,4) gix(90)];
gi2y=[rad_loc(2,4) giy(90)];
plot(gix(1:90), giy(1:90), 'b')
plot(gix(1:50), -giy(1:50)+2*rad_loc(2,4), 'b')
plot(gi1x, gi1y, 'b')
plot(gi2x, gi2y, 'b')

                                  %%%% BROOKLYN NAVAL YARD
bnyx=linspace(rad_loc(1,5)-rad_radius(5), rad_loc(1,5)+rad_radius(5));
bnyy=abs(sqrt(rad_radius(5)^2-(bnyx-rad_loc(1,5)).^2) + rad_loc(2,5));
bnyx1=[bnyx(15) rad_loc(1,5)];
bnyy1=[-bnyy(15)+2*rad_loc(2,5) rad_loc(2,5)];
bnyx2=[rad_loc(1,5) bnyx(65)];
bnyy2=[rad_loc(2,5) bnyy(65)];
plot(bnyx(1:65), bnyy(1:65), 'g')
plot(bnyx(1:15), -bnyy(1:15)+2*rad_loc(2,5), 'g')
plot(bnyx1, bnyy1, 'g')
plot(bnyx2, bnyy2, 'g')

                                  %%%% MARINERS HARBOR
mhx=linspace(rad_loc(1,2)-rad_radius(2), rad_loc(1,2)+rad_radius(2));
mhy=abs(sqrt(rad_radius(2)^2-(mhx-rad_loc(1,2)).^2) + rad_loc(2,2));
mhx1=[rad_loc(1,2) mhx(51)];
```

```
mhy1=[rad_loc(2,2) mhy(51)];
mhx2=[rad_loc(1,2) mhx(100)];
mhy2=[rad_loc(2,2) mhy(100)];
plot(mhx(51:100), mhy(51:100), 'm')
plot(mhx1, mhy1, 'm')
plot(mhx2, mhy2, 'm')
text(4,3,'Sandy')
text(4.2,2, 'Hook')
text(-4,13,'Mariners')
text(-4,12, 'Harbor')
text(2.5,13,'Bank')
text(2.5,12.5, 'Street')
text(7,16,'Governors')
text(7, 15.5, 'Island')
text(9.1,17.5,'Brooklyn')
text(9, 17, 'Naval Yard')
plot(0,0,'rX')
text(-4,0, 'lat 40 25')
text(-4, -1, 'lon 74 10')
plot(15, 0, 'rX')
text(15.3,0, 'lat 40 25')
text(15.3, -1, 'lon 73 50 21')
plot(0, 20, 'rX')
text(-4,20, 'lat 40 45')
text(-4,19, 'lon 74 10')
xlabel('Nautical Miles')
ylabel('Nautical Miles')
y=1;
```

LIST OF REFERENCES

1. Young, W., "What Are Vessel Traffic Services and What Can They Really Do?," *Navigation: Journal of The Institute of Navigation*, vol. 41, no. 1, pp. 31-53, 1994

2. U.S. Congress, House of Representatives, "Coast Guards Vessel Traffic Services 2000," Serial No. 103-52, Washington: Government Printing Office, July 29, 1993

3. Department of Transportation and U.S. Department of Defense, "1992 Federal Radionavigation Plan," Washington: Government Printing Office, 1992

4. Range Directorate Chesapeake Test Range, "System Design Document For The Coast Guard Vessel Traffic Service System," MIPR Number DTCG23-92-F-TAC111, Naval Air Warfare Center, Maryland, March, 1994

5. Waltz, E. and Llinas, J., *Multi-Sensor Data Fusion*, Artech House Inc., Boston, MA, 1990

6. Hall, David L., *Mathematical Techniques in Multisensor Data Fusion*, Artech House Inc., Norwood, MA, 1992

7. Blackman, S., *Multiple-Target Tracking with Radar Applications*, Artech House Inc., Boston, MA, 1986

8. Kandel, A., *Fuzzy Mathematical Techniques with Applications*, Addison - Wesley, Reading, MA, 1986

9. Titus, H., EC2300 Class Notes, Naval Postgraduate School, Winter, 1994

10. Kwakernaak, H. and Sivan, R., *Linear Optimal Control Systems*, Wiley-Interscience, New York, NY, 1972

11. Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1992

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer engineering Naval Postgraduate School Monterey, California 93943-5121	1
4.	Professor Murali Tummala, Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	5
5.	Professor Roberto Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	1
6.	LCDR Michael Linzey P.O. Box 60 USCG EECEN Wildwood Crest, New Jersey 08260-0060	1
7.	LT John Wood P.O. Box 60 USCG EECEN Wildwood Crest, New Jersey 08260-0060	1
8.	LT Thomas M. Ruthenberg 7948 Hemphill Dr. San Diego, California 92126	1